Guide du développeur d'ACTIONSCRIPT® 3.0

Informations juridiques

Vous trouverez des informations juridiques à l'adresse http://help.adobe.com/fr_FR/legalnotices/index.html.

Dernière mise à jour le 12/12/2014

Sommaire

Chapitre 24 : Utilisation du son

Chapitre 25 : Utilisation de la vidéo

Chapitre 26 : Utilisation de caméras

Chapitre 27 : Utilisation de la gestion des droits d'auteur numériques (DRM)

Chapitre 59 : Programmation HTML et JavaScript dans AIR

Chapitre 60 : Programmation du conteneur HTML d'AIR à l'aide de scripts

Chapitre 61 : Gestion des événements HTML dans AIR

Chapitre 62 : Affichage de contenu HTML dans une application mobile

Chapitre 63 : Utilisation de programmes de travail à des fins de simultanéité

Chapitre 64 : Sécurité

Chapitre 1 : Utilisation des dates et des heures

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si la ponctualité ne fait pas tout, elle n'en est pas moins un facteur clé des applications logicielles. ActionScript 3.0 propose différentes méthodes de gestion des dates calendaires, des heures et des intervalles temporels. Deux classes principales assurent l'essentiel de la fonctionnalité temporelle : la classe Date et la nouvelle classe Timer du package flash.utils.

Les dates et les heures constituent un type d'informations courant utilisé dans les programmes ActionScript. Par exemple, vous pouvez, entre autres, connaître la date du jour ou calculer combien de temps un utilisateur passe sur un écran particulier. Dans ActionScript, vous pouvez utiliser la classe Date pour représenter un moment unique dans le temps, y compris des informations de date et d'heure. Une occurrence de Date contient des valeurs pour les unités de date et d'heure individuelles (année, mois, date, jour de la semaine, heure, minutes, secondes, millisecondes et fuseau horaire, par exemple). Pour des utilisations plus avancées, ActionScript comprend également la classe Timer que vous pouvez utiliser pour effectuer des actions après un certain délai ou à des intervalles répétés.

Voir aussi

Date

flash.utils.Timer

Gestion des dates calendaires et des heures

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 3.0, toutes les fonctions de gestion des dates calendaires et des heures se concentrent autour de la classe de niveau supérieur Date. Cette classe contient des méthodes et des propriétés qui vous permettent de manier les dates et les heures soit selon le modèle universel coordonné (UTC, Universal Time Coordinated), soit selon le fuseau horaire considéré. L'UTC est une définition normalisée du temps qui correspond essentiellement à l'heure du méridien de Greenwich (GMT, Greenwich Mean Time).

Création d'objets Date

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Date compte l'une des méthodes constructeur les plus polyvalentes de toutes les classes de base. Vous pouvez l'invoquer de quatre manières différentes.

Primo, si aucun paramètre n'est fourni, le constructeur `Date()` renvoie un objet Date contenant la date et l'heure locale actuelle de votre fuseau horaire. Exemple :

```
var now:Date = new Date();
```

Secundo, si un paramètre numérique unique est fourni, le constructeur `Date()` le considère comme le nombre de millisecondes écoulées depuis le 1er janvier 1970 et renvoie l'objet Date correspondant. Notez que la valeur en millisecondes que vous transmettez est considérée comme le nombre de millisecondes depuis le 1er janvier 1970 en temps UTC. Toutefois, l'objet Date affiche des valeurs dans votre fuseau horaire local, sauf si vous utilisez des méthodes UTC uniquement pour les extraire et les afficher. Si vous créez un nouvel objet Date à l'aide du paramètre en millisecondes, veillez à tenir compte du décalage horaire entre votre fuseau local et le temps UTC. Les instructions ci-après créent un objet Date défini à minuit le 1er janvier 1970 en temps UTC :

```
var millisecondsPerDay:int = 1000 * 60 * 60 * 24;
// gets a Date one day after the start date of 1/1/1970
var startTime:Date = new Date(millisecondsPerDay);
```

Tertio, vous pouvez transmettre plusieurs paramètres numériques au constructeur `Date()`. Celui-ci les traite respectivement comme la date, le mois, le jour, les heures, les minutes, les secondes et les millisecondes, et renvoie un objet Date correspondant. Les paramètres indiqués sont considérés comme correspondant à l'heure locale plutôt qu'au temps UTC. Les instructions suivantes établissent un objet Date défini à minuit le 1er janvier 2000, dans le fuseau horaire local :

```
var millenium:Date = new Date(2000, 0, 1, 0, 0, 0, 0);
```

Quarto, vous pouvez transmettre un paramètre numérique unique au constructeur `Date()`. Celui-ci essaiera d'analyser cette chaîne en composants de date et heure, et de renvoyer un objet Date correspondant. Si vous choisissez cette approche, il est recommandé d'inclure le constructeur `Date()` dans un bloc `try..catch` afin d'intercepter toute erreur d'analyse. Le constructeur `Date()` gère plusieurs formats de chaîne (dont la liste figure dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash). L'instruction suivante initialise un nouvel objet Date à l'aide d'une valeur chaîne :

```
var nextDay:Date = new Date("Mon May 1 2006 11:30:00 AM");
```

Si le constructeur `Date()` ne peut analyser le paramètre chaîne, il ne déclenche pas d'exception. Cependant, l'objet Date résultant contiendra une valeur date incorrecte.

Obtention des valeurs d'unités temporelles

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez extraire les valeurs de plusieurs unités temporelles au sein de l'objet Date grâce à des propriétés ou des méthodes de la classe Date. Chacune des propriétés suivantes fournit la valeur d'une unité temporelle de l'objet Date :

- La propriété `fullYear`
- La propriété `month` au format numérique : de 0 pour janvier à 11 pour décembre
- La propriété `date`, qui correspond au numéro calendaire du jour du mois, de 1 à 31
- La propriété `day`, qui correspond au jour de la semaine au format numérique, 0 représentant dimanche
- La propriété `hours`, de 0 à 23
- La propriété `minutes`
- La propriété `seconds`
- La propriété `milliseconds`

  La classe Date offre en fait plusieurs manières d'obtenir ces valeurs. Vous pouvez par exemple obtenir la valeur month de l'objet Date de quatre manières :

- La propriété `month`

- La méthode `getMonth()`
- La propriété `monthUTC`
- La méthode `getMonthUTC()`

  Ces quatre solutions sont d'une efficacité équivalente, vous pouvez donc adopter celle qui convient le mieux à votre application.

  Les propriétés répertoriées ci-dessus représentent toutes des composants de la valeur date totale. Par exemple, la propriété milliseconds ne sera jamais supérieure à 999, puisque si elle atteint 1000, la valeur seconds augmente de 1, la propriété milliseconds se remet à zéro.

  Si vous voulez obtenir la valeur de l'objet Date en millisecondes depuis le 1er janvier 1970 (UTC), vous pouvez utiliser la méthode `getTime()`. La méthode `setTime()`, quant à elle, vous permet de modifier la valeur d'un objet Date existant en millisecondes depuis le 1er janvier 1970 (UTC).

Opérations arithmétiques sur la date et l'heure

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Date permet d'additionner et de soustraire des dates et heures. Les valeurs Date sont conservées en interne sous forme de millisecondes. Il est donc nécessaire de convertir les autres valeurs en millisecondes avant de les ajouter ou de les soustraire aux objets Date.

Si votre application doit effectuer de nombreuses opérations arithmétiques sur les dates et heures, il peut s'avérer utile de créer des constantes qui conservent les valeurs d'unités temporelles courantes en millisecondes, comme illustré ci-après :

```
public static const millisecondsPerMinute:int = 1000 * 60;
public static const millisecondsPerHour:int = 1000 * 60 * 60;
public static const millisecondsPerDay:int = 1000 * 60 * 60 * 24;
```

Il devient alors simple d'effectuer des opérations arithmétiques à l'aide des unités temporelles standard. Le code ci-après décale la valeur date d'une heure par rapport à l'heure actuelle à l'aide des méthodes `getTime()` et `setTime()` :

```
var oneHourFromNow:Date = new Date();
oneHourFromNow.setTime(oneHourFromNow.getTime() + millisecondsPerHour);
```

Une autre manière de définir une valeur date consiste à créer un objet Date à l'aide d'un seul paramètre en millisecondes. Par exemple, le code suivant ajoute 30 jours à une date pour en calculer une autre :

```
// sets the invoice date to today's date
var invoiceDate:Date = new Date();

// adds 30 days to get the due date
var dueDate:Date = new Date(invoiceDate.getTime() + (30 * millisecondsPerDay));
```

La constante `millisecondsPerDay` est ensuite multipliée par 30 pour représenter la période de 30 jours et le résultat est ajouté à la valeur `invoiceDate` afin de définir la valeur `dueDate`.

Conversion entre plusieurs fuseaux horaires

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les opérations arithmétiques sur les dates et heures sont particulièrement pratiques lorsque vous devez convertir des dates d'un fuseau horaire à un autre. Tel est le rôle de la méthode `getTimezoneOffset()`, qui renvoie la valeur (en minutes) de décalage entre le fuseau horaire de l'objet Date et le temps UTC. La valeur renvoyée s'exprime en minutes parce que tous les fuseaux horaires ne correspondent pas à une heure ; certains sont décalés d'une demi-heure par rapport aux fuseaux voisins.

L'exemple suivant utilise le décalage de fuseau horaire pour convertir une date à partir de l'heure locale en temps UTC. La conversion s'effectue tout d'abord par calcul de la valeur du fuseau horaire en millisecondes, puis par ajustement de la valeur Date selon ce montant :

```
// creates a Date in local time
var nextDay:Date = new Date("Mon May 1 2006 11:30:00 AM");

// converts the Date to UTC by adding or subtracting the time zone offset
var offsetMilliseconds:Number = nextDay.getTimezoneOffset() * 60 * 1000;
nextDay.setTime(nextDay.getTime() + offsetMilliseconds);
```

Contrôle des intervalles temporels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous développez des applications à l'aide d'Adobe Flash CS4 Professional, vous avez accès au scénario, qui fournit une progression constante, image par image, au sein de votre application. Toutefois, dans un projet purement ActionScript, vous devez compter sur d'autres mécanismes temporels.

Boucles ou minuteurs ?

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans certains langages de programmation, vous devez mettre au point des motifs temporels à l'aide d'instructions en boucle telles que `for` ou `do..while`.

Les instructions en boucle s'exécutent en général aussi vite que la machine locale le permet, ce qui signifie que l'application sera plus rapide sur certaines machines que sur d'autres. Si votre application doit bénéficier d'un intervalle temporel cohérent, vous devez l'associer à un calendrier ou une horloge réels. Bien des applications, telles que les jeux, les animations, les contrôleurs en temps réel, nécessitent des mécanismes de décompte temporel qui soient cohérents d'une machine à l'autre.

La classe Timer d'ActionScript 3.0 offre une solution performante dans ce domaine. Grâce au modèle d'événements ActionScript 3.0, la classe Timer distribue des événements Timer dès qu'un intervalle spécifié est atteint.

La classe Timer

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour la gestion des fonctions temporelles dans ActionScript 3.0, il est recommandé d'utiliser la classe Timer (flash.utils.Timer), qui permet de distribuer des événements dès qu'un intervalle est atteint.

Pour lancer un minuteur, vous devez d'abord créer une occurrence de la classe Timer et lui indiquer à quelle fréquence elle doit générer un événement Timer et combien de fois elle doit le faire avant de s'arrêter.

Par exemple, le code suivant crée une occurrence de Timer qui distribue un événement toutes les secondes pendant 60 secondes :

```
var oneMinuteTimer:Timer = new Timer(1000, 60);
```

L'objet Timer distribue un objet TimerEvent chaque fois que l'intervalle donné est atteint. Le type d'événement de l'objet TimerEvent est `timer` (défini par la constante `TimerEvent.TIMER`). Un objet TimerEvent contient les mêmes propriétés que l'objet standard Event.

Si l'occurrence de Timer prévoit un nombre fixe d'intervalles, elle distribue également un événement `timerComplete` (défini par la constante `TimerEvent.TIMER_COMPLETE`) lorsqu'elle atteint l'intervalle final.

Voici un court exemple d'application illustrant la classe Timer en action :

```
package
{
    import flash.display.Sprite;
    import flash.events.TimerEvent;
    import flash.utils.Timer;

    public class ShortTimer extends Sprite
    {
        public function ShortTimer()
        {
            // creates a new five-second Timer
            var minuteTimer:Timer = new Timer(1000, 5);

            // designates listeners for the interval and completion events
            minuteTimer.addEventListener(TimerEvent.TIMER, onTick);
            minuteTimer.addEventListener(TimerEvent.TIMER_COMPLETE, onTimerComplete);

            // starts the timer ticking
            minuteTimer.start();
        }

        public function onTick(event:TimerEvent):void
        {
            // displays the tick count so far
            // The target of this event is the Timer instance itself.
            trace("tick " + event.target.currentCount);
        }

        public function onTimerComplete(event:TimerEvent):void
        {
            trace("Time's Up!");
        }
    }
}
```

Lorsque la classe ShortTimer est créée, elle génère une occurrence de Timer qui marque chaque seconde pendant cinq secondes. Elle ajoute alors deux écouteurs au minuteur : un qui écoute chaque décompte et un qui écoute l'événement `timerComplete`.

Elle lance ensuite le décompte du minuteur et à partir de là, la méthode `onTick()` s'exécute toutes les secondes.

La méthode `onTick()` affiche simplement le nombre actuel de tics. Après cinq secondes, la méthode `onTimerComplete()` s'exécute et vous avertit que le temps est écoulé.

Si vous exécutez cet exemple, vous devriez voir les lignes suivantes s'afficher dans votre console ou fenêtre de suivi à raison d'une ligne par seconde :

```
tick 1
tick 2
tick 3
tick 4
tick 5
Time's Up!
```

Fonctions temporelles du package flash.utils

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 contient un certain nombre de fonctions temporelles similaires à celles qui étaient disponibles dans ActionScript 2.0. Ces fonctions sont fournies au niveau du package dans le package flash.utils et elles fonctionnent de la même manière que dans ActionScript 2.0.

Fonction	Description
`clearInterval(id:uint):void`	Annule un appel de `setInterval()` spécifié.
`clearTimeout(id:uint):void`	Annule un appel de `setTimeout()` spécifié.
`getTimer():int`	Renvoie le nombre de millisecondes qui se sont écoulées depuis l'initialisation d'Adobe® Flash® Player ou d'Adobe® AIR™.
`setInterval(closure:Function, delay:Number, ... arguments):uint`	Exécute une fonction à fréquence définie (intervalle exprimé en millisecondes).
`setTimeout(closure:Function, delay:Number, ... arguments):uint`	Exécute une fonction spécifiée après un délai spécifié (en millisecondes).

Ces fonctions demeurent dans ActionScript 3.0 afin d'assurer la compatibilité avec les versions antérieures. Adobe ne recommande pas leur utilisation dans les nouvelles applications ActionScript 3.0. Il est en général plus simple et plus efficace d'utiliser la classe Timer.

Exemple de date et heure : horloge analogique simple

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un exemple d'horloge analogique simple illustre ces deux concepts de date et heure :

- Obtention de la date et de l'heure actuelle et extraction des valeurs heures, minutes et secondes
- Utilisation d'une horloge pour fixer le rythme d'une application

  Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application SimpleClock se trouvent dans le dossier Samples/SimpleClock. L'application se compose des fichiers suivants :

Fichier	Description
SimpleClockApp.mxml ou SimpleClockApp.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/simpleclock/SimpleClock.as	Fichier d'application principal.
com/example/programmingas3/simpleclock/AnalogClockFace.as	Dessine un cadran d'horloge rond et les aiguilles des heures, des minutes et des secondes en fonction de l'heure.

Définition de la classe SimpleClock

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si l'exemple d'horloge est simple, il est toujours judicieux de bien organiser les applications de manière à faciliter leur extension future. Dans ce but, l'application SimpleClock utilise la classe SimpleClock pour gérer les tâches de démarrage et de mesure temporelle. Elle se sert ensuite d'une autre classe, AnalogClockFace, pour l'affichage réel de l'heure.

Voici le code qui définit et initialise la classe SimpleClock (vous remarquerez que dans la version Flash, SimpleClock étend la classe Sprite à la place) :

```
public class SimpleClock extends UIComponent
{
    /**
     * The time display component.
     */
    private var face:AnalogClockFace;

    /**
     * The Timer that acts like a heartbeat for the application.
     */
    private var ticker:Timer;
```

Cette classe possède deux propriétés importantes :

- La propriété `face`, qui correspond à une occurrence de la classe AnalogClockFace
- La propriété `ticker`, qui est une occurrence de la classe Timer

  La classe SimpleClock utilise un constructeur par défaut. La méthode `initClock()` se charge de la véritable configuration, en créant le cadran et en lançant le décompte de l'occurrence de Timer.

Création du cadran

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les lignes suivantes du code SimpleClock créent le cadran utilisé pour afficher l'heure :

```
/**
 * Sets up a SimpleClock instance.
 */
public function initClock(faceSize:Number = 200)
{
    // creates the clock face and adds it to the display list
    face = new AnalogClockFace(Math.max(20, faceSize));
    face.init();
    addChild(face);

    // draws the initial clock display
    face.draw();
```

La taille de l'horloge peut être transmise à la méthode `initClock()`. Si aucune valeur `faceSize` n'est transmise, la taille par défaut de 200 pixels est utilisée.

L'application initialise ensuite l'horloge et l'ajoute à la liste d'affichage à l'aide de la méthode `addChild()` héritée de la classe DisplayObjectContainer. Elle appelle enfin la méthode `AnalogClockFace.draw()` pour afficher une fois le cadran, qui indique l'heure actuelle.

Lancement du minuteur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une fois le cadran créé, la méthode `initClock()` définit le minuteur :

```
    // creates a Timer that fires an event once per second
    ticker = new Timer(1000);

    // designates the onTick() method to handle Timer events
    ticker.addEventListener(TimerEvent.TIMER, onTick);

    // starts the clock ticking
    ticker.start();
```

Cette méthode commence par instancier une occurrence de Timer qui va distribuer un événement par seconde (toutes les 1 000 millisecondes). Comme le constructeur `Timer()` ne reçoit pas de second paramètre `repeatCount`, l'horloge se reproduit indéfiniment.

La méthode `SimpleClock.onTick()` s'exécute une fois par seconde après réception de l'événement `timer` :

```
public function onTick(event:TimerEvent):void
{
    // updates the clock display
    face.draw();
}
```

La méthode `AnalogClockFace.draw()` dessine simplement le cadran de l'horloge et des aiguilles.

Affichage de l'heure actuelle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La plupart du code de la classe AnalogClockFace implique la définition des éléments d'affichage du cadran. Lors de son initialisation, AnalogClockFace dessine un contour circulaire, place des libellés numériques pour chaque heure, puis crée trois objets Shape, un pour l'aiguille des heures, un pour celle des minutes et un pour l'aiguille des secondes de l'horloge.

Lorsque l'application SimpleClock s'exécute, elle appelle la méthode `AnalogClockFace.draw()` toutes les secondes, comme suit :

```
/**
 * Called by the parent container when the display is being drawn.
 */
public override function draw():void
{
    // stores the current date and time in an instance variable
    currentTime = new Date();
    showTime(currentTime);
}
```

Cette méthode enregistre l'heure actuelle dans une variable, pour que l'heure ne puisse changer pendant le dessin des aiguilles de l'horloge. Elle appelle ensuite la méthode `showTime()` pour afficher les aiguilles, comme illustré ci-après :

```
/**
 * Displays the given Date/Time in that good old analog clock style.
 */
public function showTime(time:Date):void
{
    // gets the time values
    var seconds:uint = time.getSeconds();
    var minutes:uint = time.getMinutes();
    var hours:uint = time.getHours();

    // multiplies by 6 to get degrees
    this.secondHand.rotation = 180 + (seconds * 6);
    this.minuteHand.rotation = 180 + (minutes * 6);

    // Multiply by 30 to get basic degrees, then
    // add up to 29.5 degrees (59 * 0.5)
    // to account for the minutes.
    this.hourHand.rotation = 180 + (hours * 30) + (minutes * 0.5);
}
```

Tout d'abord, cette méthode extrait les valeurs des heures, des minutes et des secondes pour l'heure actuelle. Elle utilise ensuite ces valeurs pour calculer l'angle de chaque aiguille. Comme l'aiguille des secondes effectue une rotation complète en 60 secondes, elle tourne de 6 degrés par seconde (360/60). L'aiguille des minutes pivote selon le même angle chaque minute.

L'aiguille des heures se met à jour toutes les minutes également et doit donc progresser à chaque minute. Elle tourne de 30 degrés toutes les heures (360/12), mais pivote également d'un demi-degré toutes les minutes (30 degrés divisés par 60 minutes).

Chapitre 2 : Utilisation des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe String contient des méthodes qui vous permettent de manipuler des chaînes de texte. Les chaînes s'utilisent avec de nombreux objets. Les méthodes décrites ci-après permettent de manipuler les chaînes utilisées dans des objets tels que TextField, StaticText, XML, ContextMenu et FileReference.

Les chaînes sont des séries de caractères. ActionScript 3.0 prend en charge les caractères ASCII et Unicode.

Voir aussi

String

RegExp

parseFloat()

parseInt()

Principes de base des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour les programmeurs, une « chaîne » est un texte, une suite de lettres, chiffres ou autres caractères qui se suivent et forment une unité représentée par une valeur. Par exemple, la ligne de code suivante crée une variable ayant le type de données String (chaîne) et affecte une valeur de chaîne littérale à cette variable :

```
var albumName:String = "Three for the money";
```

Comme le montre cet exemple, ActionScript permet de délimiter une valeur chaîne en enfermant du texte entre des guillemets droits doubles ou simples. Voici d'autres exemples de chaînes :

```
"Hello"
"555-7649"
"http://www.adobe.com/"
```

Lorsque vous manipulez un fragment de texte en ActionScript, vous utilisez une valeur chaîne. La classe String d'ActionScript est le type de données qui permet de travailler avec du texte. Des occurrences de chaînes sont fréquemment utilisées pour des propriétés, des paramètres de méthodes, etc., dans de nombreuses autres classes en ActionScript.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs aux chaînes :

ASCII Système de codage permettant de représenter du texte sous forme de caractères et symboles dans les programmes informatiques. Le système ASCII gère les 26 lettres de l'alphabet latin, plus un nombre limité de caractères supplémentaires.

Caractère Unité de base d'un texte (lettre ou symbole).

Concaténation Ajout bout à bout de plusieurs valeurs de chaîne pour en créer une nouvelle.

Chaîne vide Chaîne qui ne contient rien: ni texte, ni espace, ni autre caractère, représentée par `""`. Une chaîne vide n'est pas la même chose qu'une variable ayant une valeur nulle (null) : celle-ci est une variable à laquelle aucune occurrence de l'objet String n'est affectée, alors qu'une chaîne vide est une occurrence dont la valeur ne contient aucun caractère.

String Valeur textuelle (séquence de caractères).

Littéral (ou « littéral de chaîne ») Valeur chaîne écrite explicitement en code, sous forme de valeur texte encadrée par des guillemets droits simples ou doubles.

Sous-chaîne Définit une chaîne qui fait partie d'une autre chaîne.

Unicode Système standardisé de codage permettant de représenter du texte sous forme de caractères et symboles dans les programmes informatiques. Le système Unicode permet d'utiliser la totalité des caractères de toutes les langues écrites existantes.

Création de chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe String permet de représenter des données de chaîne (texte) en ActionScript 3.0. Les chaînes ActionScript prennent en charge les caractères ASCII et Unicode. La meilleure façon de créer une chaîne est d'utiliser un littéral de chaîne. Pour déclarer un littéral de chaîne, utilisez les guillemets droits doubles (`"`) ou les guillemets droits simples (`'`). Par exemple, les deux chaînes suivantes sont équivalentes :

```
var str1:String = "hello";
var str2:String = 'hello';
```

Vous pouvez également déclarer une chaîne à l'aide de l'opérateur `new`, comme suit :

```
var str1:String = new String("hello");
var str2:String = new String(str1);
var str3:String = new String();       // str3 == ""
```

Les deux chaînes suivantes sont équivalentes :

```
var str1:String = "hello";
var str2:String = new String("hello");
```

Pour utiliser des guillemets droits simples (`'`) dans un littéral de chaîne délimité par des guillemets droits simples (`'`), utilisez le caractère d'échappement (`\`). De même, pour utiliser des guillemets droits doubles (`''`) dans un littéral de chaîne délimité par des guillemets droits doubles (`''`), utilisez le caractère d'échappement (`\`). Les deux chaînes suivantes sont équivalentes :

```
var str1:String = "That's \"A-OK\"";
var str2:String = 'That\'s "A-OK"';
```

Vous pouvez utiliser des guillemets simples ou des guillemets doubles en fonction de ceux qui existent dans un littéral de chaîne, comme dans l'exemple suivant :

```
var str1:String = "ActionScript <span class='heavy'>3.0</span>";
var str2:String = '<item id="155">banana</item>';
```

Rappelons qu'ActionScript fait la distinction entre les guillemets droits simples (`'`) et les guillemets simples gauches ou droits (`' ou ' `). Il en est de même pour les guillemets doubles. Utilisez des guillemets droits pour délimiter des littéraux de chaîne. Lorsque vous collez du texte dans ActionScript depuis une autre source, utilisez les caractères corrects.

Comme indiqué dans le tableau suivant, vous pouvez utiliser le caractère d'échappement (\) pour définir d'autres caractères dans des littéraux de chaîne :

Séquence d'échappement	**Caractère**
\b	Retour arrière
\f	Changement de page
\n	Nouvelle ligne
\r	Retour chariot
\t	Tabulation
\u*nnnn*	Caractère Unicode dont le code de caractère est spécifié par le nombre hexadécimal *nnnn* ; par exemple, \u263a est le caractère smiley.
\\x*nn*	Caractère ASCII dont le code est spécifié par le nombre hexadécimal *nn*.
\'	Guillemet droit simple
\"	Guillemet droit double
\\	Barre oblique inverse

Propriété length

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque chaîne possède une propriété `length` correspondant au nombre de caractères qu'elle contient:

```
var str:String = "Adobe";
trace(str.length);            // output: 5
```

Une chaîne vide et une chaîne null ont toutes deux une longueur de 0, comme l'indique l'exemple suivant :

```
var str1:String = new String();
trace(str1.length);           // output: 0

str2:String = '';
trace(str2.length);           // output: 0
```

Utilisation de caractères dans des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque caractère d'une chaîne possède une position d'index dans la chaîne (un entier). La position d'index du premier caractère est 0. Par exemple, dans la chaîne suivante, le caractère `y` occupe la position 0 et le caractère `w` occupe la position 5 :

```
"yellow"
```

Vous pouvez examiner des caractères individuels à différentes positions d'une chaîne à l'aide des méthodes `charAt()` et `charCodeAt()`, comme dans l'exemple suivant :

```
var str:String = "hello world!";
for (var i:int = 0; i < str.length; i++)
{
    trace(str.charAt(i), "-", str.charCodeAt(i));
}
```

Lorsque vous exécutez ce code, vous obtenez le résultat suivant :

```
h - 104
e - 101
l - 108
l - 108
o - 111
- 32
w - 119
o - 111
r - 114
l - 108
d - 100
! - 33
```

Vous pouvez également utiliser des codes de caractère pour définir une chaîne à l'aide de la méthode `fromCharCode()`, comme l'indique l'exemple suivant :

```
var myStr:String = String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
        // Sets myStr to "hello world!"
```

Comparaison de chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser les opérateurs suivants pour comparer des chaînes : `<`, `<=`, `!=`, `==`, `=>` et `>`. Vous pouvez utiliser ces opérateurs avec des instructions conditionnelles (`if` et `while`, par exemple) comme l'indique l'exemple suivant :

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2)
{
    trace("A < a, B < b, C < c, ...");
}
```

Lorsque vous utilisez ces opérateurs avec des chaînes, ActionScript considère la valeur du code de chaque caractère dans la chaîne, en comparant les caractères de gauche à droite, comme indiqué ci-dessous :

```
trace("A" < "B"); // true
trace("A" < "a"); // true
trace("Ab" < "az"); // true
trace("abc" < "abza"); // true
```

Utilisez les opérateurs `==` et `!=` pour comparer des chaînes entre elles et pour comparer des chaînes avec d'autres types d'objets, comme l'indique l'exemple suivant :

```
var str1:String = "1";
var str1b:String = "1";
var str2:String = "2";
trace(str1 == str1b); // true
trace(str1 == str2); // false
var total:uint = 1;
trace(str1 == total); // true
```

Récupération des représentations de chaîne d'autres objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez obtenir une représentation de chaîne de n'importe quel type d'objet. Tous les objets disposent en effet d'une méthode `toString()` :

```
var n:Number = 99.47;
var str:String = n.toString();
    // str == "99.47"
```

Lorsque vous utilisez l'opérateur de concaténation + avec une combinaison d'objet String et d'objets qui ne sont pas des chaînes, vous n'avez pas besoin d'utiliser la méthode `toString()`. Pour plus d'informations sur la concaténation, voir la section suivante.

La fonction globale `String()` renvoie la même valeur pour un objet donné que la valeur renvoyée par l'objet appelant la méthode `toString()`.

Concaténation de chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La concaténation de chaînes consiste à prendre deux chaînes et à les combiner en une seule chaîne de façon séquentielle. Par exemple, vous pouvez utiliser l'opérateur + pour concaténer deux chaînes :

```
var str1:String = "green";
var str2:String = "ish";
var str3:String = str1 + str2; // str3 == "greenish"
```

Vous pouvez également utiliser l'opérateur += pour obtenir le même résultat, comme dans l'exemple suivant :

```
var str:String = "green";
str += "ish"; // str == "greenish"
```

En outre, la classe String comprend la méthode `concat()`, utilisable comme suit :

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
// str4 == "Bonjour from Paris"
```

Si vous utilisez l'opérateur + (ou l'opérateur +=) avec un objet String et un objet qui n'est *pas* une chaîne, ActionScript convertit automatiquement ce dernier en un objet String afin d'évaluer l'expression, comme indiqué dans l'exemple :

```
var str:String = "Area = ";
var area:Number = Math.PI * Math.pow(3, 2);
str = str + area; // str == "Area = 28.274333882308138"
```

Néanmoins, vous pouvez utiliser des parenthèses pour le regroupement afin de fournir un contexte à l'opérateur +, comme indiqué dans l'exemple suivant :

```
trace("Total: $" + 4.55 + 1.45); // output: Total: $4.551.45
trace("Total: $" + (4.55 + 1.45)); // output: Total: $6
```

Recherche de sous-chaînes et de modèles dans des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les sous-chaînes sont des caractères consécutifs à l'intérieur d'une chaîne. La chaîne `"abc"`, par exemple, contient les sous-chaînes suivantes : `""`, `"a"`, `"ab"`, `"abc"`, `"b"`, `"bc"`, `"c"`. Vous pouvez utiliser des méthodes ActionScript pour localiser les sous-chaînes d'une chaîne.

Les modèles sont définis en ActionScript par des chaînes ou par des expressions régulières. Par exemple, l'expression régulière suivante définit un modèle spécifique, les lettres A, B, et C suivies d'un chiffre (les barres de fraction sont des délimiteurs d'expression régulière) :

```
/ABC\d/
```

ActionScript comporte des méthodes servant à rechercher des modèles dans des chaînes et à remplacer les correspondances trouvées par des sous-chaînes de substitution. Ces méthodes sont décrites dans les sections suivantes.

Les expressions régulières peuvent définir des modèles compliqués. Pour plus d'informations, voir « Utilisation d'expressions régulières » à la page 78.

Recherche d'une sous-chaîne par la position d'un caractère

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes `substr()` et `substring()` sont similaires. Elles renvoient toutes les deux une sous-chaîne d'une chaîne. Elles prennent deux paramètres. Dans les deux méthodes, le premier paramètre est la position du caractère initial dans la chaîne concernée. Toutefois, dans la méthode `substr()`, le deuxième paramètre est la *longueur* de la sous-chaîne à renvoyer, alors que dans la méthode `substring()`, le deuxième paramètre est la position du caractère *final* de la sous-chaîne (qui ne figure pas dans la chaîne renvoyée). Cet exemple illustre la différence entre ces deux méthodes :

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.substr(11,15)); // output: Paris, Texas!!!
trace(str.substring(11,15)); // output: Pari
```

La méthode `slice()` fonctionne de la même façon que la méthode `substring()`. Lorsque deux nombres entiers non négatifs sont passés en paramètres, son fonctionnement est identique. Néanmoins, la méthode `slice()` peut recevoir des entiers négatifs comme paramètres. Dans ce cas, la position des caractères est comptée à partir de la fin de la chaîne, comme dans l'exemple suivant :

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.slice(11,15)); // output: Pari
trace(str.slice(-3,-1)); // output: !!
trace(str.slice(-3,26)); // output: !!!
trace(str.slice(-3,str.length)); // output: !!!
trace(str.slice(-8,-3)); // output: Texas
```

Vous pouvez associer des entiers positifs et négatifs comme paramètres de la méthode `slice()`.

Recherche de la position des caractères d'une sous-chaîne correspondante

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser les méthodes `indexOf()` et `lastIndexOf()` pour localiser des sous-chaînes correspondantes au sein d'une chaîne, comme dans l'exemple suivant :

```
var str:String = "The moon, the stars, the sea, the land";
trace(str.indexOf("the")); // output: 10
```

La méthode `indexOf()` est sensible à la casse.

Vous pouvez spécifier un deuxième paramètre pour indiquer la position de l'index dans la chaîne à partir de laquelle commencer la recherche, comme suit :

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.indexOf("the", 11)); // output: 21
```

La méthode `lastIndexOf()` trouve la dernière occurrence d'une sous-chaîne dans la chaîne :

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.lastIndexOf("the")); // output: 30
```

Si vous incluez un deuxième paramètre avec la méthode `lastIndexOf()`, la recherche est effectuée à l'envers à partir de cette position d'index dans la chaîne (de droite à gauche) :

```
var str:String = "The moon, the stars, the sea, the land"
trace(str.lastIndexOf("the", 29)); // output: 21
```

Création d'un tableau de sous-chaînes segmenté par un délimiteur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la méthode `split()` pour créer un tableau de sous-chaînes divisé en fonction d'un caractère délimiteur. Par exemple, vous pouvez segmenter une chaîne séparée par des virgules ou des tabulations en plusieurs chaînes.

L'exemple suivant indique comment diviser un tableau en sous-chaînes avec le caractère esperluette (&) comme délimiteur :

```
var queryStr:String = "first=joe&last=cheng&title=manager&StartDate=3/6/65";
var params:Array = queryStr.split("&", 2); // params == ["first=joe","last=cheng"]
```

Le deuxième paramètre de la méthode `split()` (qui est facultatif) définit la taille maximale du tableau renvoyé.

Vous pouvez également utiliser une expression régulière comme caractère délimiteur :

```
var str:String = "Give me\t5."
var a:Array = str.split(/\s+/); // a == ["Give","me","5."]
```

Pour plus d'informations, voir « Utilisation d'expressions régulières » à la page 78 et le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Recherche de modèles dans des chaînes et remplacement de sous-chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe String comprend les méthodes suivantes pour utiliser des modèles dans des chaînes :

- Utilisez les méthodes `match()` et `search()` pour localiser des sous-chaînes qui correspondent à un modèle.
- Utilisez la méthode `replace()` pour rechercher des sous-chaînes qui correspondent à un modèle et les remplacer par une sous-chaîne spécifiée.

Ces méthodes sont décrites dans les sections suivantes.

Vous pouvez utiliser des chaînes ou des expressions régulières pour définir des modèles utilisés dans ces méthodes. Pour plus d'informations sur les expressions régulières, voir « Utilisation d'expressions régulières » à la page 78.

Recherche de sous-chaînes correspondantes

La méthode `search()` renvoie la position de l'index de la première sous-chaîne qui correspond à un modèle donné, comme illustré dans cet exemple :

```
var str:String = "The more the merrier.";
// (This search is case-sensitive.)
trace(str.search("the")); // output: 9
```

Vous pouvez également utiliser des expressions régulières pour définir le modèle pour lequel établir une correspondance, comme illustré dans cet exemple :

```
var pattern:RegExp = /the/i;
var str:String = "The more the merrier.";
trace(str.search(pattern)); // 0
```

Le résultat de la méthode `trace()` est 0, car le premier caractère dans la chaîne est la position de l'index 0. L'indicateur `i` est défini dans l'expression régulière. Par conséquent, la recherche n'est pas sensible à la casse.

La méthode `search()` trouve une seule correspondance et renvoie sa position d'index de début, même si l'indicateur `g` (global) est défini dans l'expression régulière.

L'exemple suivant présente une expression régulière plus compliquée qui correspond à une chaîne dans des guillemets doubles :

```
var pattern:RegExp = /"[^"]*"/;
var str:String = "The \"more\" the merrier.";
trace(str.search(pattern)); // output: 4

str = "The \"more the merrier.";
trace(str.search(pattern)); // output: -1
// (Indicates no match, since there is no closing double quotation mark.)
```

La méthode `match()` fonctionne de façon similaire. Elle recherche une sous-chaîne correspondante. Néanmoins, lorsque vous utilisez l'indicateur global dans un modèle d'expression régulière (comme dans l'exemple suivant), `match()` renvoie un tableau de sous-chaînes correspondantes :

```
var str:String = "bob@example.com, omar@example.org";
var pattern:RegExp = /\w*@\w*\.[org|com]+/g;
var results:Array = str.match(pattern);
```

Le tableau `results` est défini comme suit :

```
["bob@example.com","omar@example.org"]
```

Pour plus d'informations sur les expressions régulières, voir « Utilisation d'expressions régulières » à la page 78.

Remplacement de sous-chaînes mises en correspondance

Vous pouvez utiliser la méthode `replace()` pour rechercher un modèle spécifié dans une chaîne et remplacer les correspondances par la chaîne de remplacement spécifiée, comme illustré dans l'exemple suivant :

```
var str:String = "She sells seashells by the seashore.";
var pattern:RegExp = /sh/gi;
trace(str.replace(pattern, "sch")); //sche sells seaschells by the seaschore.
```

Dans cet exemple, les chaînes mises en correspondance ne sont pas sensibles à la casse car l'indicateur `i` (`ignoreCase`) est défini dans l'expression régulière, et plusieurs correspondances sont remplacées car l'indicateur `g` (`global`) est défini. Pour plus d'informations, voir « Utilisation d'expressions régulières » à la page 78.

Vous pouvez inclure les codes de remplacement `$` suivants dans la chaîne de remplacement. Le texte de remplacement indiqué dans le tableau suivant est inséré à la place du code de remplacement `$` :

Code $	Texte de remplacement
`$$`	$
`$&`	Sous-chaîne correspondante.
`$``	Partie de la chaîne qui précède la sous-chaîne correspondante. Ce code utilise les guillemets simples droits gauches (`) et non les guillemets simples droits (') ni les guillemets simples anglais gauches (').
`$'`	Partie de la chaîne qui suit la sous-chaîne correspondante. Ce code utilise les guillemets simples droits (').
$n	*n*ième groupe entre parenthèses correspondant capturé, où n est un chiffre compris entre 1 et 9 et $n n'est pas suivi d'une décimale.
$nn	*nn*ième groupe entre parenthèses correspondant capturé, où *nn* est un nombre décimal à deux chiffres compris entre 01 et 99. Si la *nn*ième capture n'est pas définie, le texte de remplacement est une chaîne vide.

L'exemple suivant illustre l'utilisation des codes de remplacement `$2` et `$1`, qui représentent le premier et le deuxième groupes capturés correspondants :

```
var str:String = "flip-flop";
var pattern:RegExp = /(\w+)-(\w+)/g;
trace(str.replace(pattern, "$2-$1")); // flop-flip
```

Vous pouvez également utiliser une fonction comme deuxième paramètre de la méthode `replace()`. Le texte correspondant est remplacé par la valeur renvoyée de la fonction.

```
var str:String = "Now only $9.95!";
var price:RegExp = /\$([\d,]+.\d+)+/i;
trace(str.replace(price, usdToEuro));

function usdToEuro(matchedSubstring:String,  capturedMatch1:String,  index:int,
str:String):String
{
    var usd:String = capturedMatch1;
    usd = usd.replace(",", "");
    var exchangeRate:Number = 0.853690;
    var euro:Number = parseFloat(usd) * exchangeRate;
    const euroSymbol:String = String.fromCharCode(8364);
    return euro.toFixed(2) + " " + euroSymbol;
}
```

Lorsque vous utilisez une fonction comme deuxième paramètre de la méthode `replace()`, les arguments suivants sont transmis à la fonction :

- La partie correspondante de la chaîne.
- Tout groupe entre parenthèses capturé correspondant. Le nombre d'arguments transmis de cette façon varie selon le nombre de correspondances entre parenthèses. Pour déterminer le nombre de correspondances entre parenthèses, vérifiez `arguments.length - 3` dans le code de la fonction.
- La position d'index dans la chaîne où débute la correspondance.
- La chaîne complète.

Conversion de la casse dans des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme illustré dans l'exemple suivant, les méthodes `toLowerCase()` et `toUpperCase()` convertissent les caractères alphabétiques de la chaîne en minuscule et en majuscule, respectivement :

```
var str:String = "Dr. Bob Roberts, #9."
trace(str.toLowerCase()); // dr. bob roberts, #9.
trace(str.toUpperCase()); // DR. BOB ROBERTS, #9.
```

Une fois que ces méthodes sont exécutées, la chaîne source reste inchangée. Pour transformer la chaîne source, utilisez le code suivant :

```
str = str.toUpperCase();
```

Ces méthodes fonctionnent avec des caractères étendus, pas simplement a–z et A–Z :

```
var str:String = "José Barça";
trace(str.toUpperCase(), str.toLowerCase()); // JOSÉ BARÇA josé barça
```

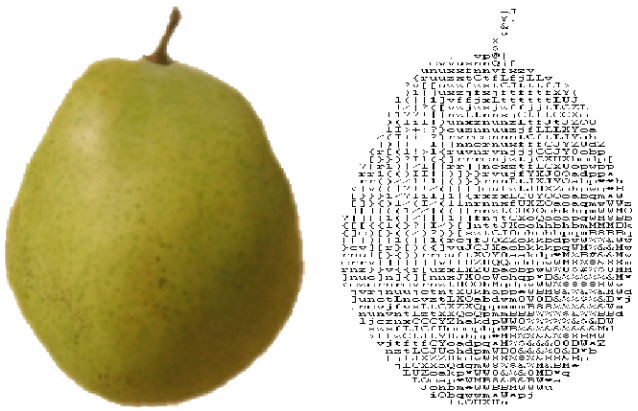
Exemple de chaîne : ASCII Art

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple ASCII Art représente différentes façons d'utiliser la classe String en ActionScript 3.0, notamment :

- La méthode `split()` de la classe String est utilisée pour extraire des valeurs d'une chaîne séparée par des caractères (informations d'image dans un fichier de texte séparé par des tabulations).
- Plusieurs techniques de manipulation de chaînes, y compris `split()`, la concaténation et l'extraction d'une partie de la chaîne à l'aide de `substring()` et de `substr()`, sont utilisées pour mettre la première lettre de chaque mot dans les titres d'image en majuscule.
- La méthode `getCharAt()` est utilisée pour obtenir un seul caractère à partir d'une chaîne (pour déterminer le caractère ASCII correspondant à une valeur bitmap d'échelle de gris).
- La concaténation de chaîne est utilisée pour construire la représentation ASCII art d'une image, un caractère à la fois.

Le terme *ASCII art* fait référence à des représentations textuelles d'une image dans lesquelles une grille de polices de caractères à espacement constant (caractères Courier New, par exemple) trace l'image. L'image suivante est un exemple d'ASCII art produit par l'application :



La version ASCII art du graphique est illustrée à droite.

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application ASCIIArt se trouvent dans le dossier Samples/AsciiArt. L'application se compose des fichiers suivants :

Fichier	Description
AsciiArtApp.mxml ou AsciiArtApp.fla	Fichier d'application principal en FLA pour Flash ou en MXML pour Flex
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	La classe qui fournit la fonctionnalité principale de l'application, notamment l'extraction de métadonnées d'image d'un fichier de texte, le chargement des images et la gestion du processus de conversion d'image en texte.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	Une classe qui fournit la méthode `parseBitmapData()` pour convertir des données d'image dans une version String.
com/example/programmingas3/asciiArt/Image.as	Une classe qui représente une image bitmap chargée.

Fichier	Description
com/example/programmingas3/asciiArt/ImageInfo.as	Une classe représentant des métadonnées pour une image ASCII art (titre, URL de fichier image, etc.).
image/	Un dossier contenant des images utilisées par l'application.
txt/ImageData.txt	Un fichier de texte séparé par des tabulations et contenant des informations sur les images à charger par l'application.

Extraction de valeurs séparées par des tabulations

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple utilise le stockage séparé de données d'application par rapport à l'application ; de cette façon, si les données changent (par exemple, si une autre image est ajoutée ou que le titre d'une image change), il est inutile de recréer le fichier SWF. Dans ce cas, les métadonnées d'image, y compris le titre de l'image, l'URL du fichier d'image réel et certaines valeurs utilisées pour manipuler l'image, sont stockés dans un fichier de texte (le fichier txt/ImageData.txt dans le projet). Le contenu du fichier de texte est le suivant :

```
FILENAMETITLEWHITE_THRESHHOLDBLACK_THRESHHOLD
FruitBasket.jpgPear, apple, orange, and bananad810
Banana.jpgA picture of a bananaC820
Orange.jpgorangeFF20
Apple.jpgpicture of an apple6E10
```

Le fichier utilise un format séparé par des tabulations spécifique. La première ligne est une ligne d'en-tête. Les lignes restantes contiennent les données suivantes pour chaque bitmap à charger :

- Le nom de fichier du bitmap.
- Le nom d'affichage du bitmap.
- Les valeurs de seuil du blanc et les valeurs de seuil du noir pour les bitmaps. Il s'agit de valeurs hexadécimales au-dessus et en dessous desquelles un pixel doit être considéré comme totalement blanc ou totalement noir.

Dès que l'application démarre, la classe AsciiArtBuilder se charge et analyse le contenu du fichier de texte afin de créer la « `pile` » d'images qu'elle chargera. Pour cela, elle utilise le code suivant de la méthode parseImageInfo() de la classe AsciiArtBuilder :

```
var lines:Array = _imageInfoLoader.data.split("\n");
var numLines:uint = lines.length;
for (var i:uint = 1; i < numLines; i++)
{
    var imageInfoRaw:String = lines[i];
    ...
    if (imageInfoRaw.length > 0)
    {
        // Create a new image info record and add it to the array of image info.
        var imageInfo:ImageInfo = new ImageInfo();

        // Split the current line into values (separated by tab (\t)
        // characters) and extract the individual properties:
        var imageProperties:Array = imageInfoRaw.split("\t");
        imageInfo.fileName = imageProperties[0];
        imageInfo.title = normalizeTitle(imageProperties[1]);
        imageInfo.whiteThreshold = parseInt(imageProperties[2], 16);
        imageInfo.blackThreshold = parseInt(imageProperties[3], 16);
        result.push(imageInfo);
    }
}
```

Le contenu entier du fichier de texte se trouve dans une seule occurrence de String, la propriété `_imageInfoLoader.data`. Vous pouvez utiliser la méthode `split()` avec le caractère de nouvelle ligne (`"\n"`) comme paramètre pour diviser l'occurrence de String en un tableau (`lines`) dont les éléments sont les lignes individuelles du fichier de texte. Le code utilise ensuite une boucle pour travailler avec chacune des lignes (excepté la première car elle contient uniquement des en-têtes). A l'intérieur de la boucle, la méthode `split()` est de nouveau utilisée pour diviser le contenu de la ligne en un ensemble de valeurs (l'objet Array appelé `imageProperties`). Le paramètre utilisé avec la méthode `split()` dans ce cas est le caractère de tabulation (`"\t"`) car les valeurs dans chaque ligne sont délimitées par des tabulations.

Utilisation de méthodes String pour normaliser des titres d'image

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans cette application, tous les titres d'image sont affichés à l'aide d'un format standard, avec la première lettre de chaque mot en majuscule (excepté quelques mots qui ne sont généralement pas en majuscule dans des titres anglais). Au lieu de considérer que le fichier de texte contient des titres formatés correctement, l'application formate les titres lors de leur extraction du fichier de texte.

Dans la liste de code précédente, lors de l'extraction de valeurs de métadonnées d'image individuelles, la ligne de code suivante est utilisée :

```
imageInfo.title = normalizeTitle(imageProperties[1]);
```

Dans ce code, le titre de l'image issu du fichier de texte est transmis au moyen de la méthode `normalizeTitle()` avant d'être stocké dans l'objet ImageInfo :

```
private function normalizeTitle(title:String):String
{
    var words:Array = title.split(" ");
    var len:uint = words.length;
    for (var i:uint; i < len; i++)
    {
        words[i] = capitalizeFirstLetter(words[i]);
    }

    return words.join(" ");
}
```

Cette méthode utilise la méthode `split()` pour diviser le titre en mots individuels (séparés par le caractère d'espacement), transmet chaque mot au moyen de la méthode `capitalizeFirstLetter()` puis utilise la méthode `join()` de la classe Array pour combiner de nouveau les mots en une chaîne unique.

Comme son nom l'indique, la méthode `capitalizeFirstLetter()` met la première lettre de chaque mot en majuscule :

```
/**
 * Capitalizes the first letter of a single word, unless it's one of
 * a set of words that are normally not capitalized in English.
 */
private function capitalizeFirstLetter(word:String):String
{
    switch (word)
    {
        case "and":
        case "the":
        case "in":
        case "an":
        case "or":
        case "at":
        case "of":
        case "a":
            // Don't do anything to these words.
            break;
        default:
            // For any other word, capitalize the first character.
            var firstLetter:String = word.substr(0, 1);
            firstLetter = firstLetter.toUpperCase();
            var otherLetters:String = word.substring(1);
            word = firstLetter + otherLetters;
    }
    return word;
}
```

En anglais, le premier caractère des mots suivants utilisés dans un titre n'est *pas* mis en majuscule : "and," "the," "in," "an," "or," "at," "of," ou "a." (il s'agit d'une version simplifiée des règles). Pour exécuter cette logique, le code utilise d'abord une instruction `switch` pour vérifier si le mot est l'un des mots ne devant pas être en majuscule. Si c'est le cas, le code sort de l'instruction `switch`. Si le mot doit être en majuscule, la procédure comprend plusieurs étapes :

1. La première lettre du mot est extraite à l'aide de `substr(0, 1)`, qui extraie une sous-chaîne commençant par le caractère au niveau de l'index 0 (la première lettre de la chaîne, comme indiqué par le premier paramètre `0`). La sous-chaîne contiendra un caractère (indiqué par le deuxième paramètre `1`).
2. Ce caractère est mis en majuscule à l'aide de la méthode `toUpperCase()`.

3 Les caractères restants du mot d'origine sont extraits à l'aide de `substring(1)`, qui extraie une sous-chaîne commençant à l'index 1 (la deuxième lettre) jusqu'à la fin de la chaîne (indiqué en omettant le deuxième paramètre de la méthode `substring()`).

4 Le dernier mot est créé en combinant la première lettre mise en majuscule et les lettres restantes en utilisant la concaténation de chaîne : `firstLetter + otherLetters`

Génération du texte ASCII art

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe BitmapToAsciiConverter permet de convertir une image bitmap en sa représentation de texte ASCII. Cette procédure est effectuée par la méthode `parseBitmapData()`, partiellement représentée ici :

```
var result:String = "";

// Loop through the rows of pixels top to bottom:
for (var y:uint = 0; y < _data.height; y += verticalResolution)
{
    // Within each row, loop through pixels left to right:
    for (var x:uint = 0; x < _data.width; x += horizontalResolution)
    {
        ...

        // Convert the gray value in the 0-255 range to a value
        // in the 0-64 range (since that's the number of "shades of
        // gray" in the set of available characters):
        index = Math.floor(grayVal / 4);
        result += palette.charAt(index);
    }
    result += "\n";
}
return result;
```

Ce code définit d'abord une occurrence de String appelée `result` qui sera utilisée pour créer la version ASCII art de l'image bitmap. Il effectue ensuite une boucle sur les pixels de l'image bitmap source. Il utilise plusieurs techniques de manipulation des couleurs (non décrites ici) pour convertir le rouge, le vert et le bleu d'un pixel en une valeur d'échelle de gris (un nombre entre 0 et 255). Le code divise ensuite cette valeur par 4 (comme indiqué) pour la convertir en une valeur dans l'échelle 0-63, qui est stockée dans la variable `index` (l'échelle 0-63 est utilisée car la palette des caractères ASCII disponibles utilisée par cette application contient 64 valeurs). La palette des caractères est définie en tant qu'occurrence de String dans la classe BitmapToAsciiConverter :

```
// The characters are in order from darkest to lightest, so that their
// position (index) in the string corresponds to a relative color value
// (0 = black).
private static const palette:String =
"@#$%&8BMW*mwqpdbkhaoQ0OZXYUJCLtfjzxnuvcr[]{}1()|/?Il!i><+_~-;,. ";
```

Etant donné que la variable `index` définit le caractère ASCII de la palette qui correspond au pixel actuel dans l'image bitmap, ce caractère est récupéré de la `palette` String à l'aide de la méthode `charAt()`. Il est ensuite ajouté à l'occurrence de String `result` au moyen de l'opérateur d'affectation de concaténation (`+=`). En outre, à la fin de chaque ligne de pixels, un caractère de nouvelle ligne est concaténé à la fin de l'occurrence de String `result` afin que la ligne à renvoyer crée une ligne de pixels de caractères.

Chapitre 3 : Utilisation de tableaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les tableaux vous permettent de stocker plusieurs valeurs dans une seule structure de données. Vous pouvez utiliser des tableaux indexés simples qui stockent des valeurs à l'aide d'index d'entiers ordinaux fixes ou des tableaux associatifs complexes qui stockent des valeurs à l'aide de clés arbitraires. Les tableaux peuvent également être multidimensionnels et contenir des éléments étant eux-mêmes des tableaux. Pour finir, vous pouvez utiliser un vecteur pour les tableaux dont les éléments sont tous des occurrences du même type de données.

Voir aussi

Array

Vecteur

Principes de base des tableaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous aurez souvent besoin en programmation d'utiliser un ensemble d'éléments plutôt qu'un seul objet; par exemple, dans une application de lecteur de musique, vous pouvez avoir une liste de morceaux en attente de lecture. Vous ne souhaitez pas créer une variable séparée pour chaque morceau de cette liste. Il serait préférable de rassembler tous les objets Song et de les utiliser sous forme de groupe.

Un tableau est un élément de programmation qui agit comme conteneur pour un ensemble d'éléments (une liste de morceaux, par exemple). La plupart du temps, tous les éléments d'un tableau sont des occurrences de la même classe, mais ceci n'est pas obligatoire dans ActionScript. Les éléments individuels d'un tableau sont les *éléments* du tableau. Un tableau peut être comparé à un tiroir classeur pour des variables. Les variables peuvent être ajoutées en tant qu'éléments au tableau, comme vous placez un dossier dans le tiroir classeur. Vous pouvez utiliser le tableau comme une variable unique, comme si vous transportiez le tiroir entier à un autre endroit. Vous pouvez utiliser les variables en tant que groupe, comme si vous recherchiez des informations dans les dossiers en les parcourant l'un après l'autre. Vous pouvez y accéder individuellement, comme si vous ouvriez le tiroir et sélectionniez un seul dossier.

Par exemple, imaginez que vous créez une application de lecteur de musique dans laquelle un utilisateur peut sélectionner plusieurs morceaux et les ajouter à une liste de lecture. Dans votre code ActionScript, vous avez une méthode appelée `addSongsToPlaylist()` qui accepte un seul tableau comme paramètre. Peu importe le nombre de morceaux à ajouter à la liste (quelques-uns, un grand nombre, ou même un seul), vous devez appeler la méthode `addSongsToPlaylist()` une seule fois, en lui transmettant le tableau qui contient les objets Song. Dans la méthode `addSongsToPlaylist()`, vous pouvez utiliser une boucle pour parcourir les éléments (morceaux) du tableau un par un et les ajouter à la liste de lecture.

Le type de tableau ActionScript le plus courant est le *tableau indexé*. Dans un tableau indexé, chaque élément est stocké dans un emplacement numéroté appelé *index*. On accède à des éléments à l'aide du numéro, comme une adresse. Les tableaux indexés répondent à la plupart des besoins en programmation. La classe Array est une classe courante utilisée pour représenter un tableau indexé.

Un tableau indexé est souvent utilisé pour stocker plusieurs éléments du même type, des objets qui sont des occurrences de la même classe. La classe Array ne dispose pas de moyens pour restreindre le type d'éléments qu'elle contient. La classe Vector est un type de tableau indexé dans lequel tous les éléments d'un tableau unique sont du même type. L'utilisation d'une occurrence de Vector à la place d'une occurrence de Array peut également déboucher sur une amélioration des performances entre autres. la classe Vector est prise en charge à partir de Flash Player 10 et Adobe AIR 1.5.

Une utilisation spéciale d'un tableau indexé est un *tableau multidimensionnel*. Un tableau multidimensionnel est un tableau indexé dont les éléments sont des tableaux indexés, qui contiennent à leur tour d'autres éléments.

Le *tableau associatif* est un autre type de tableau. Il utilise une chaîne *key* au lieu d'un index numérique pour identifier des éléments individuels. Enfin, ActionScript 3.0 comprend également une classe Dictionary qui représente un *dictionnaire*. Un dictionnaire est un tableau qui vous permet d'utiliser tout type d'objet comme clé afin de distinguer les éléments entre eux.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants utilisés dans le cadre de la programmation de routines qui gèrent des tableaux et des vecteurs :

Array Objet qui sert de conteneur pour regrouper plusieurs objets.

Opérateur ([]) d'accès au tableau Paire de crochets entourant un index ou une clé qui identifie de façon unique un élément du tableau. Cette syntaxe est utilisée après le nom d'une variable de tableau pour spécifier un seul élément du tableau plutôt qu'un tableau entier.

Tableau associatif Tableau qui utilise des clés de chaîne pour identifier des éléments individuels.

Type de base Type de données des objets qu'une occurrence de Vector est autorisée à stocker.

Dictionnaire Tableau dont les éléments sont constitués de paires d'objets appelées clé et valeur. La clé est utilisée à la place d'un index numérique pour identifier un seul élément.

Elément Elément unique dans un tableau.

Index Adresse numérique utilisée pour identifier un élément unique dans un tableau indexé.

Tableau indexé Type de tableau standard qui stocke chaque élément dans une position numérotée et utilise le numéro (index) pour identifier des éléments individuels.

Clé Chaîne ou objet utilisé pour identifier un seul élément dans un tableau associatif ou un dictionnaire.

Tableau multidimensionnel Tableau contenant des éléments qui sont des tableaux plutôt que des valeurs uniques.

T Convention standard utilisée dans la présente documentation pour représenter le type de base d'une occurrence de Vector, quel qu'il soit. La convention T est utilisée pour représenter un nom de classe, comme cela est indiqué dans la description du paramètre Type. (« T » correspond à « type », comme dans « type de données ».)

Paramètre Type Syntaxe utilisée avec le nom de classe Vector pour spécifier le type de base de Vector (le type de données des objets qu'il stocke). La syntaxe consiste en un point (.), puis le nom du type de données entouré de parenthèses en chevron (<>). L'ensemble ressemble à ceci : `Vector<T>`. Dans la présente documentation, la classe spécifiée dans le paramètre Type est représenté de façon générique par `T`.

Vecteur Type de tableau dont les éléments sont tous des occurrences du même type de données.

Tableaux indexés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les tableaux indexés stockent une série d'une ou de plusieurs valeurs organisées de façon à ce que vous puissiez accéder à chaque valeur à l'aide d'une valeur d'entier non signé. Le premier index correspond toujours au nombre 0. L'index est ensuite incrémenté d'une unité pour chaque élément ajouté consécutivement au tableau. Dans ActionScript 3.0, deux classes sont utilisées comme tableaux indexés : la classe Array et la classe Vector.

Les tableaux indexés utilisent un entier 32 bits non signé pour le numéro d'index. La taille maximale d'un tableau indexé est 2^{32} - 1 ou 4 294 967 295. Si vous tentez de créer un tableau plus grand que la taille maximale, une erreur d'exécution se produit.

Pour accéder à un élément particulier d'un tableau indexé, vous pouvez utiliser l'opérateur (`[]`) d'accès au tableau pour spécifier la position de l'index de l'élément visé. Par exemple, le code suivant représente le premier élément (l'élément à l'index 0) dans un tableau indexé appelé `songTitles` :

```
songTitles[0]
```

La combinaison du nom de la variable du tableau suivi de l'index entre crochets fonctionne comme un identifiant unique. En d'autres termes, elle peut être utilisée tout comme un nom de variable. Vous pouvez affecter une valeur à un élément du tableau indexé en utilisant le nom et l'index du côté gauche d'une instruction d'affectation :

```
songTitles[1] = "Symphony No. 5 in D minor";
```

Dans la même veine, vous pouvez récupérer une valeur à un élément du tableau indexé en utilisant le nom et l'index du côté droit d'une instruction d'affectation :

```
var nextSong:String = songTitles[2];
```

Vous pouvez aussi utiliser une variable entre crochets plutôt que de fournir une valeur explicite. Elle doit contenir une valeur entière non-négative telle qu'un uint, un int positif ou une occurrence de Number d'entier positif. Cette technique est utilisée couramment pour passer en boucle sur les éléments dans un tableau indexé et effectuer une opération sur un ou tous les éléments. Le code ci-dessous décrit cette technique. Le code utilise une boucle pour accéder à chaque valeur dans un objet Array appelé `oddNumbers`. Il utilise l'instruction `trace()` pour imprimer chaque valeur sous la forme "oddNumber[*index*] = *value*" :

```
var oddNumbers:Array = [1, 3, 5, 7, 9, 11];
var len:uint = oddNumbers.length;
for (var i:uint = 0; i < len; i++)
{
    trace("oddNumbers[" + i.toString() + "] = " + oddNumbers[i].toString());
}
```

Classe Array

La classe Array est le premier type de tableau indexé. Une occurrence de Array peut comporter une valeur de n'importe quel type de données. Le même objet Array peut comporter des objets qui sont de types de données différents. Par exemple, une occurrence de Array unique peut avoir une valeur String en index 0, une occurrence de Number en index 1 et un objet XML en index 2.

Classe Vector

La classe Vector est un autre type de tableau indexé qui est disponible dans ActionScript 3.0. Une occurrence de Vector est un *tableau typé*, ce qui signifie que tous les éléments d'une occurrence de Vector ont toujours le même type de données.

Remarque : *la classe Vector est prise en charge à partir de Flash Player 10 et Adobe AIR 1.5.*

Lorsque vous déclarez une variable Vector ou que vous instanciez un objet Vector, vous spécifiez explicitement le type de données des objets que le vecteur peut contenir. Le type de données spécifié est connu sous le nom de *type de base* du vecteur. Lors de l'exécution et de la compilation (en mode strict), tout code qui fixe la valeur d'un élément Vector ou récupère une valeur d'un élément Vector est contrôlé. Si le type de données de l'objet que l'on tente d'ajouter ou de récupérer ne correspond pas au type de base du vecteur, une erreur se produit.

Outre la restriction concernant le type de données, la classe Vector possède d'autres restrictions qui la distinguent de la classe Array :

- Un vecteur est un tableau dense. Un objet Array peut comporter des valeurs dans les index 0 et 7 même si elle n'en a pas dans les positions 1 à 6. Cependant, un vecteur doit comporter une valeur (ou `null`) dans chaque index.
- Un vecteur peut facultativement avoir une longueur fixe. Ceci signifie que le nombre d'éléments du vecteur est immuable.
- L'accès aux éléments d'un vecteur est défini par ses limites. Vous ne pouvez jamais lire une valeur d'un index supérieur à celui de l'élément final (`longueur` - 1). Vous ne pouvez jamais définir une valeur avec un index supérieur à l'index final actuel. En d'autres termes, vous pouvez définir une valeur uniquement à l'index existant ou à une `[longueur]` d'index.

En raison de ses restrictions, un vecteur présente trois avantages principaux par rapport à une occurrence d'Array dont les éléments sont tous des occurrences d'une classe unique :

- Performance : l'accès à l'élément de tableau et son itération sont beaucoup plus rapides lorsque vous utilisez une occurrence de Vector que lorsque vous utilisez une occurrence d'Array.
- Sécurité des types : en mode strict, le compilateur peut identifier les erreurs de type de données. Parmi ces erreurs, il y a l'affectation d'une valeur du type de données incorrect à un vecteur ou l'attente d'un type de données incompatible lors de la lecture d'une valeur à partir du vecteur. A l'exécution, les types de données sont également contrôlés lors de l'ajout de données à un objet Vector ou la lecture de données qui en provient. Notez cependant que lorsque vous utilisez la méthode `push()` ou `unshift()` pour ajouter des valeurs à un vecteur, les types de données des arguments ne sont pas vérifiés au moment de la compilation. Lorsque vous utilisez ces méthodes, les valeurs sont toujours contrôlées à l'exécution.
- Fiabilité : le contrôle de gamme à l'exécution (ou contrôle de longueur fixe) assure une fiabilité nettement supérieure à celle des objets Array.

A part les contraintes et les avantages supplémentaires, la classe Vector est très proche de la classe Array. Les propriétés et les méthodes d'un objet Vector sont similaires, voire dans certains cas identiques, aux propriétés et aux méthodes d'un objet Array. Au lieu d'utiliser un objet Array dont tous les éléments possèdent le même type de données, il est généralement préférable de faire appel à une occurrence de l'objet Vector.

Création de tableaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser plusieurs techniques pour créer une occurrence de Array ou une occurrence de Vector. Cependant, les techniques de création de chaque type de tableau sont quelque peu différentes.

Création d'une occurrence de Array

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous créez un objet Array par l'appel au constructeur `Array(` ou par l'utilisation d'une syntaxe de littéral de tableau.

Vous pouvez utiliser la fonction de constructeur `Array()` de trois façons différentes. Premièrement, si vous appelez le constructeur sans arguments, vous obtenez un tableau vide. Vous pouvez utiliser la propriété `length` de la classe Array pour vérifier que le tableau ne contient aucun élément. Par exemple, le code suivant appelle le constructeur `Array()` sans arguments :

```
var names:Array = new Array();
trace(names.length); // output: 0
```

Deuxièmement, si vous utilisez un nombre comme unique paramètre pour le constructeur `Array()`, un tableau de cette longueur est créé, avec chaque valeur d'élément définie sur `undefined`. L'argument doit être un entier non signé compris entre les valeurs 0 et 4 294 967 295. Par exemple, le code suivant appelle le constructeur `Array()` avec un seul argument numérique :

```
var names:Array = new Array(3);
trace(names.length); // output: 3
trace(names[0]); // output: undefined
trace(names[1]); // output: undefined
trace(names[2]); // output: undefined
```

Troisièmement, si vous appelez le constructeur et transmettez une liste d'éléments comme paramètres, un tableau est créé avec des éléments correspondant à chacun des paramètres. Le code suivant transmet trois arguments au constructeur `Array()` :

```
var names:Array = new Array("John", "Jane", "David");
trace(names.length); // output: 3
trace(names[0]); // output: John
trace(names[1]); // output: Jane
trace(names[2]); // output: David
```

Vous pouvez aussi créer des tableaux avec des littéraux de tableau. Un littéral de tableau peut être affecté directement à une variable de tableau, comme illustré dans l'exemple suivant :

```
var names:Array = ["John", "Jane", "David"];
```

Création d'une occurrence de Vector

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous créez une occurrence de Vector par l'appel du constructeur `Vector.<T>()`. Vous pouvez aussi créer un vecteur par l'appel à la fonction globale `Vector.<T>()`. Cette fonction convertit un objet spécifié en une occurrence de Vector. Dans Flash Professional CS5 et les versions ultérieures, Flash Builder 4 et les versions ultérieures et Flex 4 et les versions ultérieures, vous pouvez également créer une occurrence de Vector à l'aide de la syntaxe de littéral correspondante.

Toutes les fois que vous déclarez une variable Vector (ou de la même façon, un paramètre de la méthode Vector ou un type de renvoi de la méthode Vector), vous spécifiez le type de base de la variable Vector. Vous spécifiez également le type de base lorsque vous créez une occurrence de Vector par l'appel au constructeur `Vector.<T>()`. Autrement dit, toutes les fois que vous utilisez le terme `Vector` dans ActionScript, il est accompagné d'un type de base.

Vous spécifiez le type de base du vecteur à l'aide de la syntaxe de paramètres de type. Le paramètre de type suit immédiatement le mot `Vector` dans le code. Il est formé d'un point (`.`), puis du nom de classe de base entouré de parenthèses en chevron (`<>`), comme l'indique cet exemple :

```
var v:Vector.<String>;
v = new Vector.<String>();
```

Dans la première ligne de cet exemple, la variable `v` est déclarée comme une occurrence de `Vector.<Chaîne>`. En d'autres termes, il représente un tableau indexé qui ne peut comporter que des occurrences de String. La deuxième ligne appelle le constructeur `Vector()` pour créer une occurrence du même type Vector, c'est-à-dire un vecteur dont les éléments sont tous des objets String. Il affecte cet objet à `v`.

Utilisation du constructeur Vector.<T>()

Si vous utilisez le constructeur `Vector.<T>()` sans arguments, il crée une occurrence de Vector vide. Vous pouvez contrôler qu'un vecteur est vide en vérifiant sa propriété `length`. Par exemple, le code ci-dessus appelle le constructeur `Vector.<T>()` sans arguments :

```
var names:Vector.<String> = new Vector.<String>();
trace(names.length); // output: 0
```

Si vous savez d'avance de combien d'éléments un vecteur a besoin initialement, vous pouvez prédéfinir ce nombre dans le vecteur. Pour créer un vecteur avec un certain nombre d'éléments, transmettez le nombre d'éléments comme premier paramètre (le paramètre `length`). Comme les éléments du vecteur ne peuvent pas être vides, ils sont remplis d'occurrences du type de base. Si ce type est un type de référence qui autorise les valeurs `null`, les éléments contiennent tous `null`. Autrement, ils contiennent tous la valeur par défaut pour la classe. Par exemple, une variable uint ne peut pas être `null`. Par conséquent, dans le code ci-dessous, le vecteur appelé `ages` est créé avec sept éléments, chacun contenant la valeur 0.

```
var ages:Vector.<uint> = new Vector.<uint>(7);
trace(ages); // output: 0,0,0,0,0,0,0
```

Enfin, à l'aide du constructeur `Vector.<T>()`, vous pouvez également créer un vecteur de longueur fixe en transmettant `true` comme deuxième paramètre (le paramètre `fixed`). Dans ce cas, le vecteur est créé avec le nombre spécifié d'éléments et celui-ci ne peut pas être modifié. Notez cependant que vous pouvez quand même changer les valeurs des éléments d'un vecteur de longueur fixe.

Utilisation du constructeur de syntaxe de littéral Vector

Dans Flash Professional CS5 et les versions ultérieures, Flash Builder 4 et les versions ultérieures et Flex 4 et les versions ultérieures, vous pouvez transmettre une liste de valeurs au constructeur `Vector<T>()` pour stipuler les valeurs initiales du vecteur :

```
// var v:Vector.<T> = new <T>[E0, ..., En-1 ,];
// For example:
var v:Vector.<int> = new <int>[0,1,2,];
```

Cette syntaxe possède les caractéristiques suivantes :

- La virgule de fin de ligne est facultative.
- La syntaxe ne gère pas la présence d'éléments vides dans le tableau. Une instruction telle que `var v:Vector.<int> = new <int>[0,,2,]` renvoie une erreur de compilation.
- Il est impossible de stipuler la longueur par défaut de l'occurrence de Vector. La longueur correspond au nombre d'éléments qui composent la liste d'initialisation.
- Il est impossible de spécifier si l'occurrence de Vector possède une longueur fixe. Utilisez à cet effet la propriété `fixed`.
- Il risque de se produire des pertes de données ou des erreurs si les éléments transmis en tant que valeurs ne correspondent pas au type indiqué. Exemple :

  ```
  var v:Vector.<int> = new <int>[4.2]; // compiler error when running in strict mode
  trace(v[0]); //returns 4 when not running in strict mode
  ```

Utilisation du constructeur Vector. <T>()

Outre le constructeur `Vector.<T>()` et le constructeur de syntaxe de littéral Vector, vous disposez également de la fonction globale `Vector. <T>()` pour créer un objet Vector. La fonction globale `Vector.<T>()` est une fonction de conversion. Lorsque vous appelez la fonction globale `Vector.<T>()`, vous spécifiez le type de base du vecteur que la méthode renvoie. Vous transmettez un tableau indexé unique (occurrence de Array ou Vector) comme argument. La méthode renvoie alors un vecteur avec le type de base spécifié, contenant les valeurs dans l'argument du tableau source. Le code ci-dessous montre la syntaxe nécessaire pour appeler la fonction globale `Vector.<T>()`.

```
var friends:Vector.<String> = Vector.<String>(["Bob", "Larry", "Sarah"]);
```

La fonction globale `Vector.<T>()` exécute une conversion de type de base sur deux niveaux. D'abord, lorsqu'une occurrence de Array est transmise à la fonction, une occurrence de Vector est renvoyée. Ensuite, que le tableau source soit une occurrence de Array ou Vector, la fonction tente de convertir les éléments du tableau source en valeurs du type de base. La conversion utilise des règles standard de conversion des types de données ActionScript. Par exemple, le code suivant convertit les valeurs String du tableau source en nombres entiers dans le vecteur résultant. La partie décimale de la première (`"1.5"`) est tronquée et la troisième valeur non numérique (`"Waffles"`) est convertie en 0 dans le résultat :

```
var numbers:Vector.<int> = Vector.<int>(["1.5", "17", "Waffles"]);
trace(numbers); // output: 1,17,0
```

S'il n'est pas possible de convertir un élément source quelconque, une erreur se produit.

Lorsque le code appelle la fonction globale `Vector.<T>()`, si un élément du tableau source est une occurrence d'une sous-classe du type de base spécifié, l'élément est ajouté au vecteur résultant (aucune erreur ne se produit). L'utilisation de la fonction globale `Vector.<T>()` est en fait le seul moyen de convertir un vecteur avec un type de base `T` en un vecteur avec un type de base qui est une superclasse de `T`.

Insertion d'éléments de tableau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'opérateur (`[]`) d'accès au tableau constitue le moyen le plus élémentaire d'ajouter un élément à un tableau indexé. Pour définir la valeur d'un élément de tableau indexé, utilisez le nom d'objet Array ou Vector et le numéro d'index du côté gauche d'une instruction d'affectation.

```
songTitles[5] = "Happy Birthday";
```

Si un élément ne se trouve pas déjà à cette position d'index du tableau ou du vecteur, l'index est créé et la valeur y est stockée. Si une valeur existe à cet index, la nouvelle valeur remplace l'ancienne.

Un objet Array vous permet de créer un élément pour tout index. Cependant, avec un objet Vector, vous pouvez affecter uniquement une valeur à un index existant ou à l'index disponible suivant. Celui-ci correspond à la propriété `length` de l'objet Vector. Utilisez du code comme celui qui est présenté ci-dessous pour ajouter un nouvel élément à un objet Vector de la façon la plus sûre :

```
myVector[myVector.length] = valueToAdd;
```

Trois méthodes des classes Array et Vector, `push()`, `unshift()` et `splice()`, vous permettent d'insérer des éléments dans un tableau indexé. La méthode `push()` ajoute un ou plusieurs éléments à la fin d'un tableau. Ainsi, le dernier élément inséré dans le tableau à l'aide de la méthode `push()` aura le numéro d'index le plus élevé. La méthode `unshift()` insère un ou plusieurs éléments au début d'un tableau, qui est toujours au numéro d'index 0. La méthode `splice()` insère des éléments au niveau d'un index spécifié dans le tableau.

L'exemple suivant illustre les trois méthodes. Un tableau appelé `planets` est créé pour trier les noms des planètes par ordre de proximité par rapport au soleil. La méthode `push()` est tout d'abord appelée pour ajouter l'élément initial, `Mars`. Deuxièmement, la méthode `unshift()` est appelée pour insérer l'élément `Mercury` au début du tableau. Pour finir, la méthode `splice()` est appelée pour insérer les éléments `Venus` et `Earth` après `Mercury`, mais avant `Mars`. Le premier élément envoyé à `splice()`, l'entier 1, indique à l'insertion de débuter à l'index 1. Le deuxième argument envoyé à `splice()`, l'entier 0, indique qu'aucun élément ne doit être supprimé. Pour finir, les troisième et quatrième arguments envoyés à `splice()`, `Venus` et `Earth`, sont les éléments à insérer.

```
var planets:Array = new Array();
planets.push("Mars"); // array contents: Mars
planets.unshift("Mercury"); // array contents: Mercury,Mars
planets.splice(1, 0, "Venus", "Earth");
trace(planets); // array contents: Mercury,Venus,Earth,Mars
```

Les méthodes `push()` et `unshift()` renvoient toutes les deux un entier non signé qui représente la longueur du tableau modifié. La méthode `splice()` renvoie un tableau vide lorsqu'elle est utilisée pour insérer des éléments, ce qui semble étrange mais compréhensible en raison de saversatilité. Vous pouvez utiliser la méthode `splice()` non seulement pour insérer des éléments dans un tableau, mais également pour en supprimer. Lorsque vous l'utilisez pour supprimer des éléments, la méthode `splice()` renvoie un tableau contenant les éléments supprimés.

Remarque : *si une propriété `fixed` de l'objet Vector est définie sur `true`, le nombre total d'éléments du vecteur reste immuable. Si vous tentez d'ajouter un nouvel élément à un vecteur de longueur fixe à l'aide des techniques décrites ici, une erreur se produit.*

Récupération des valeurs et suppression des éléments du tableau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez l'opérateur (`[]`) d'accès au tableau pour récupérer la valeur d'un élément de la façon la plus simple. Pour récupérer la valeur d'un élément de tableau indexé, utilisez le nom d'objet Array ou Vector et le numéro d'index du côté droit d'une instruction d'affectation.

```
var myFavoriteSong:String = songTitles[3];
```

Il est possible d'essayer de récupérer une valeur à partir d'un tableau ou d'un vecteur à l'aide d'un index où aucun élément n'existe. Dans ce cas, un objet Array renvoie la valeur non définie et un vecteur renvoie une exception RangeError.

Trois méthodes des classes Array et Vector, `pop()`, `shift()` et `splice()`, vous permettent de supprimer des éléments. La méthode `pop()` supprime un élément de la fin du tableau. En d'autres termes, elle supprime l'élément au niveau du numéro d'index le plus élevé. La méthode `shift()` supprime un élément du début du tableau, ce qui signifie qu'elle supprime toujours l'élément au numéro d'index 0. La méthode `splice()`, qui peut également être utilisée pour insérer des éléments, supprime un nombre arbitraire d'éléments en commençant au numéro d'index indiqué par le premier argument envoyé à la méthode.

L'exemple suivant utilise les trois méthodes pour supprimer des éléments d'une occurrence d'Array. Un tableau nommé `oceans` est créé pour stocker les noms des océans. Certains noms dans le tableau sont des noms de lacs plutôt que des noms d'océans. Ils doivent donc être supprimés.

Premièrement, la méthode `splice()` est utilisée pour supprimer les éléments `Aral` et `Superior`, et insérer les éléments `Atlantic` et `Indian`. Le premier argument envoyé à `splice()`, l'entier 2, indique que l'opération doit commencer par le troisième élément dans la liste, qui est à l'index 2. Le deuxième argument, 2, indique que deux éléments doivent être supprimés. Les arguments restants, `Atlantic` et `Indian`, sont des valeurs à insérer à l'index 2.

Deuxièmement, la méthode `pop()` est utilisée pour supprimer le dernier élément dans le tableau, `Huron`. Et troisièmement, la méthode `shift()` est utilisée pour supprimer le premier élément dans le tableau, `Victoria`.

```
var oceans:Array = ["Victoria", "Pacific", "Aral", "Superior", "Indian", "Huron"];
oceans.splice(2, 2, "Arctic", "Atlantic"); // replaces Aral and Superior
oceans.pop(); // removes Huron
oceans.shift(); // removes Victoria
trace(oceans);// output: Pacific,Arctic,Atlantic,Indian
```

Les méthodes `pop()` et `shift()` renvoient toutes les deux l'élément qui a été supprimé. Pour une occurrence de Array, le type de données de la valeur renvoyée est Object car les tableaux peuvent contenir des valeurs de n'importe quel type de données. Pour une occurrence de Vector, le type de données de la valeur renvoyée est le type de base du vecteur. La méthode `splice()` renvoie un tableau ou un vecteur contenant les valeurs supprimées. Vous pouvez modifier l'exemple du tableau `oceans` de façon à ce que l'appel à `splice()` affecte le tableau renvoyé à une nouvelle variable de tableau, comme illustré dans l'exemple suivant :

```
var lakes:Array = oceans.splice(2, 2, "Arctic", "Atlantic");
trace(lakes); // output: Aral,Superior
```

Il se peut que vous rencontriez un code qui utilise l'opérateur `delete` sur un élément de l'objet Array. L'opérateur `delete` définit la valeur d'un élément de tableau sur `undefined`, mais il ne supprime pas l'élément du tableau. Par exemple, le code suivant utilise l'opérateur `delete` sur le troisième élément dans le tableau `oceans`, mais la longueur du tableau demeure à 5 :

```
var oceans:Array = ["Arctic", "Pacific", "Victoria", "Indian", "Atlantic"];
delete oceans[2];
trace(oceans);// output: Arctic,Pacific,,Indian,Atlantic
trace(oceans[2]); // output: undefined
trace(oceans.length); // output: 5
```

Vous pouvez tronquer un tableau ou un vecteur à l'aide d'une propriété `length` de tableau. Si vous définissez la propriété `length` d'un tableau indexé sur une longueur qui est moindre que la longueur actuelle du tableau, celui-ci est tronqué : tous les éléments stockés à des numéros d'index supérieurs à la nouvelle valeur `length`, diminuée de 1, sont supprimés. Par exemple, si le tableau `oceans` était trié de telle façon que toutes les entrées valides se trouvaient au début du tableau, vous pourriez utiliser la propriété `length` pour supprimer les entrées de fin de tableau, comme l'indique le code ci-dessous :

```
var oceans:Array = ["Arctic", "Pacific", "Victoria", "Aral", "Superior"];
oceans.length = 2;
trace(oceans); // output: Arctic,Pacific
```

Remarque : *si une propriété `fixed` de l'objet Vector est définie sur `true`, le nombre total d'éléments du vecteur reste immuable. Si vous essayez de supprimer un élément d'un vecteur de longueur fixe ou de tronquer celui-ci à l'aide des techniques décrites ici, une erreur se produit.*

Tri d'un tableau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Trois méthodes, `reverse()`, `sort()` et `sortOn()`, vous permettent de modifier l'ordre d'un tableau indexé, soit en triant, soit en inversant l'ordre. Toutes ces méthodes modifient le tableau existant. Le tableau ci-dessous résume ces méthodes et leurs comportements pour les objets Array et Vector :

Méthode	Comportement d'Array	Comportement de Vector
`reverse()`	Modifie l'ordre des éléments de telle sorte que le dernier élément devient le premier élément, le pénultième le deuxième, etc.	Identique au comportement d'Array
`sort()`	Vous permet de trier les éléments du tableau de diverses façons prédéfinies, comme l'ordre alphabétique ou numérique. Vous pouvez également spécifier un algorithme de tri personnalisé.	Trie les éléments suivant l'algorithme de tri personnalisé que vous spécifiez
`sortOn()`	Vous permet de trier des objets qui ont une ou plusieurs propriétés en commun en spécifiant la ou les propriétés à utiliser comme critères de tri.	Non disponible dans la classe Vector

Méthode reverse()

La méthode `reverse()` ne prend aucun paramètre et ne renvoie aucune valeur mais vous permet de faire basculer l'ordre de votre tableau de son état actuel à l'ordre inverse. L'exemple suivant inverse l'ordre des océans répertoriés dans le tableau `oceans` :

```
var oceans:Array = ["Arctic", "Atlantic", "Indian", "Pacific"];
oceans.reverse();
trace(oceans); // output: Pacific,Indian,Atlantic,Arctic
```

Tri de base avec la méthode sort() (classe Array uniquement)

Pour une occurrence d'Array, la méthode `sort()` réorganise les éléments dans un tableau à l'aide de l'*ordre de tri par défaut*. L'ordre de tri par défaut possède les caractéristiques suivantes :

- Le tri est sensible à la casse, ce qui signifie que les majuscules précédent les minuscules. Par exemple, la lettre D précède la lettre b.
- Le tri est croissant, ce qui signifie que les codes de caractère bas (A, par exemple) précédent les codes de caractère élevés (B, par exemple).
- Le tri place les valeurs identiques les unes à côté des autres mais sans ordre particulier.
- Le tri est basé sur des chaînes, ce qui signifie que les éléments sont convertis en chaînes avant d'être comparés (par exemple, 10 précède 3 car la chaîne `"1"` a un code de caractère inférieur à celui de la chaîne `"3"`).

Vous pouvez trier votre tableau en ignorant la casse ou par ordre décroissant. Vous pouvez également trier les nombres de votre tableau par ordre numérique plutôt que par ordre alphabétique. La méthode `sort()` de la classe Array possède un paramètre `options` qui vous permet de modifier chaque caractéristique de l'ordre de tri par défaut. Les options sont définies par un ensemble de constantes statiques dans la classe Array, comme indiqué dans la liste suivante :

- `Array.CASEINSENSITIVE` : cette option permet d'ignorer la casse lors du tri. Par exemple, la lettre minuscule b précède la lettre majuscule D.
- `Array.DESCENDING` : cette option inverse le tri croissant par défaut. Par exemple, la lettre B précède la lettre A.
- `Array.UNIQUESORT` : cette option permet d'arrêter le tri si deux valeurs identiques sont repérées.
- `Array.NUMERIC` : cette option permet d'effectuer un tri numérique, de façon à ce que 3 précède 10.

L'exemple suivant met en évidence certaines de ces options. Un tableau appelé `poets` est créé. Il est trié à l'aide de plusieurs options.

```
var poets:Array = ["Blake", "cummings", "Angelou", "Dante"];
poets.sort(); // default sort
trace(poets); // output: Angelou,Blake,Dante,cummings

poets.sort(Array.CASEINSENSITIVE);
trace(poets); // output: Angelou,Blake,cummings,Dante

poets.sort(Array.DESCENDING);
trace(poets); // output: cummings,Dante,Blake,Angelou

poets.sort(Array.DESCENDING | Array.CASEINSENSITIVE); // use two options
trace(poets); // output: Dante,cummings,Blake,Angelou
```

Tri personnalisé avec la méthode sort() (classes Array et Vector)

En plus du tri de base qui est disponible pour un objet Array, vous pouvez également établir une règle de tri personnalisée. Cette technique est la seule forme de méthode `sort()` disponible pour la classe Vector. Pour définir un tri personnalisé, vous rédigez une fonction de tri personnalisée et la transmettez comme argument à la méthode `sort()`.

Par exemple, si vous avez une liste de noms dans laquelle chaque élément de liste contient le nom entier d'une personne mais que vous souhaitez trier la liste par nom de famille, vous devez utiliser une fonction de tri personnalisé pour analyser chaque élément et utiliser le nom de famille dans la fonction de tri. Le code suivant indique comment procéder avec une fonction personnalisée utilisée comme paramètre pour la méthode `Array.sort()` :

```
var names:Array = new Array("John Q. Smith", "Jane Doe", "Mike Jones");
function orderLastName(a, b):int
{
    var lastName:RegExp = /\b\S+$/;
    var name1 = a.match(lastName);
    var name2 = b.match(lastName);
    if (name1 < name2)
    {
        return -1;
    }
    else if (name1 > name2)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
trace(names); // output: John Q. Smith,Jane Doe,Mike Jones
names.sort(orderLastName);
trace(names); // output: Jane Doe,Mike Jones,John Q. Smith
```

La fonction de tri personnalisé `orderLastName()` utilise une expression régulière pour extraire le nom de famille de chaque élément à utiliser pour l'opération de comparaison. L'identifiant de fonction `orderLastName` est l'unique paramètre utilisé lors de l'appel à la méthode `sort()` sur le tableau `names`. La fonction de tri accepte deux paramètres, `a` et `b`, car elle fonctionne sur deux éléments de tableau à la fois. La valeur renvoyée de la fonction de tri indique la manière dont les éléments doivent être triés :

- Une valeur renvoyée de -1 indique que le premier paramètre, `a`, précède le second paramètre, `b`.
- Une valeur renvoyée de 1 indique que le second paramètre, `b`, précède le premier, `a`.

- Une valeur renvoyée de 0 indique que les éléments ont une précédence de tri équivalente.

Méthode sortOn() (classe Array uniquement)

La méthode `sortOn()` est conçue pour des objets Array avec des éléments contenant des objets. Ces objets doivent avoir au moins une propriété en commun pouvant être utilisée comme clé de tri. L'utilisation de la méthode `sortOn()` pour des tableaux d'autres types provoque des résultats inattendus.

Remarque : *la classe Vector ne contient pas de méthode `sortOn()`. Cette méthode n'est disponible que pour les objets Array.*

L'exemple suivant modifie le tableau `poets` de façon à ce que chaque élément soit un objet plutôt qu'une chaîne. Chaque objet contient à la fois le nom de famille du poète et sa date de naissance.

```
var poets:Array = new Array();
poets.push({name:"Angelou", born:"1928"});
poets.push({name:"Blake", born:"1757"});
poets.push({name:"cummings", born:"1894"});
poets.push({name:"Dante", born:"1265"});
poets.push({name:"Wang", born:"701"});
```

Vous pouvez utiliser la méthode `sortOn()` pour trier le tableau par la propriété `born`. La méthode `sortOn()` définit deux paramètres, `fieldName` et `options`. L'argument `fieldName` doit être spécifié en tant que chaîne. Dans l'exemple suivant, la méthode `sortOn()` est appelée avec deux arguments, "`born`" et `Array.NUMERIC`. L'argument `Array.NUMERIC` est utilisé pour vérifier que le tri est effectué par ordre numérique plutôt que par ordre alphabétique. Ceci est utile même lorsque tous les nombres ont le même nombre de chiffres car vous êtes certain que le tri se fera comme prévu si un nombre comportant un nombre inférieur ou supérieur de chiffres est ensuite ajouté au tableau.

```
poets.sortOn("born", Array.NUMERIC);
for (var i:int = 0; i < poets.length; ++i)
{
    trace(poets[i].name, poets[i].born);
}
/* output:
Wang 701
Dante 1265
Blake 1757
cummings 1894
Angelou 1928
*/
```

Tri sans modification du tableau d'origine (classe Array uniquement)

Généralement, les méthodes `sort()` et `sortOn()` modifient un tableau. Si vous souhaitez trier un tableau sans modifier le tableau existant, transmettez la constante `Array.RETURNINDEXEDARRAY` avec le paramètre `options`. Cette option indique aux méthodes de renvoyer un nouveau tableau qui reflète le tri et laisse le tableau d'origine inchangé. Le tableau renvoyé par les méthodes est un tableau simple de numéros d'index qui reflète le nouvel ordre de tri et ne contient aucun élément du tableau d'origine. Par exemple, pour trier le tableau `poets` par année de naissance sans le modifier, incluez la constante `Array.RETURNINDEXEDARRAY` dans l'argument transmis pour le paramètre `options`.

L'exemple suivant stocke les informations d'index renvoyées dans un tableau nommé `indices` et utilise le tableau `indices` avec le tableau `poets` inchangé pour trier les poètes dans l'ordre de leur année de naissance :

```
var indices:Array;
indices = poets.sortOn("born", Array.NUMERIC | Array.RETURNINDEXEDARRAY);
for (var i:int = 0; i < indices.length; ++i)
{
    var index:int = indices[i];
    trace(poets[index].name, poets[index].born);
}
/* output:
Wang 701
Dante 1265
Blake 1757
cummings 1894
Angelou 1928
*/
```

Interrogation d'un tableau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les quatre méthodes restantes des classes Array et Vector, `concat()`, `join()`, `slice()`, `toString()`, interrogent toutes le tableau sans le modifier. Les méthodes `concat()` et `slice()` renvoient toutes les deux de nouveaux tableaux, alors que les méthodes `join()` et `toString()` renvoient des chaînes. La méthode `concat()` prend un nouveau tableau ou une liste d'éléments comme arguments et le/la combine avec le tableau existant pour créer un tableau. La méthode `slice()` possède deux paramètres nommés `startIndex` et `endIndex`, et renvoie un nouveau tableau contenant une copie des éléments découpés du tableau existant. La découpe commence avec l'élément à `startIndex` et se termine avec l'élément juste avant `endIndex`. Il convient d'insister : l'élément à `endIndex` n'est pas compris dans la valeur renvoyée.

L'exemple suivant utilise `concat()` et `slice()` pour créer des tableaux à l'aide d'éléments d'autres tableaux :

```
var array1:Array = ["alpha", "beta"];
var array2:Array = array1.concat("gamma", "delta");
trace(array2); // output: alpha,beta,gamma,delta

var array3:Array = array1.concat(array2);
trace(array3); // output: alpha,beta,alpha,beta,gamma,delta

var array4:Array = array3.slice(2,5);
trace(array4); // output: alpha,beta,gamma
```

Vous pouvez utiliser les méthodes `join()` et `toString()` pour interroger le tableau et renvoyer son contenu sous la forme d'une chaîne. Si aucun paramètre n'est utilisé pour la méthode `join()`, les deux méthodes se comportent de façon identique : elles renvoient une chaîne contenant une liste de tous les éléments du tableau, séparés par une virgule. La méthode `join()`, contrairement à la méthode `toString()`, accepte un paramètre nommé `delimiter`, qui permet de choisir le symbole à utiliser comme séparateur entre chaque élément de la chaîne renvoyée.

L'exemple suivant crée un tableau nommé `rivers` et appelle à la fois `join()` et `toString()` pour renvoyer les valeurs dans le tableau sous la forme d'une chaîne. La méthode `toString()` est utilisée pour renvoyer des valeurs séparées par une virgule (`riverCSV`), alors que la méthode `join()` est utilisée pour renvoyer des valeurs séparées par le caractère `+`.

```
var rivers:Array = ["Nile", "Amazon", "Yangtze", "Mississippi"];
var riverCSV:String = rivers.toString();
trace(riverCSV); // output: Nile,Amazon,Yangtze,Mississippi
var riverPSV:String = rivers.join("+");
trace(riverPSV); // output: Nile+Amazon+Yangtze+Mississippi
```

Il existe un problème avec la méthode `join()` ; toutes les occurrences de tableau et de vecteur imbriquées sont toujours renvoyées avec des valeurs séparées par des virgules, quel que soit le séparateur que vous spécifiez pour les éléments de tableau principaux, comme illustré dans l'exemple suivant :

```
var nested:Array = ["b","c","d"];
var letters:Array = ["a",nested,"e"];
var joined:String = letters.join("+");
trace(joined); // output: a+b,c,d+e
```

Tableaux associatifs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un tableau associatif, parfois appelé *hachage* ou *mappage*, utilise des *clés* plutôt qu'un index numérique pour organiser des valeurs stockées. Chaque clé dans un tableau associatif est une chaîne unique qui est utilisée pour accéder à une valeur stockée. Un tableau associatif est une occurrence de la classe Object, ce qui signifie que chaque clé correspond à un nom de propriété. Les tableaux associatifs sont des collections non triées de paires de clés et de valeurs. Votre code ne doit pas s'attendre à ce que les clés d'un tel tableau se présentent dans un ordre précis.

ActionScript 3.0 contient aussi un type avancé de tableau associatif appelé *dictionnaire*. Les dictionnaires, qui sont des occurrences de la classe Dictionary dans le package flash.utils, utilisent des clés de tout type de données. En d'autres termes, les clés de dictionnaire ne sont pas limitées à des valeurs de type String.

Tableaux associatifs avec clés de chaîne

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Deux méthodes permettent de créer des tableaux associatifs dans ActionScript 3.0. La première consiste à utiliser une occurrence de Object. Celle-ci vous permet d'initialiser votre tableau avec un littéral d'objet. Une occurrence de la classe Object, également appelée *objet générique*, fonctionne comme un tableau associatif. Chaque nom de propriété de l'objet générique devient la clé qui permet d'accéder à une valeur stockée.

L'exemple suivant crée un tableau associatif appelé `monitorInfo`, à l'aide d'un littéral d'objet pour initialiser le tableau avec deux paires de clés et de valeurs :

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"], monitorInfo["resolution"]);
// output: Flat Panel 1600 x 1200
```

Si vous n'avez pas besoin d'initialiser le tableau lors de la déclaration, vous pouvez utiliser le constructeur Object pour créer le tableau, comme suit :

```
var monitorInfo:Object = new Object();
```

Une fois que le tableau est créé à l'aide d'un littéral d'objet ou du constructeur de la classe Object, vous pouvez lui ajouter de nouvelles valeurs à l'aide de l'opérateur (`[]`) d'accès au tableau ou de l'opérateur point (`.`). L'exemple suivant ajoute deux nouvelles valeurs à `monitorArray` :

```
monitorInfo["aspect ratio"] = "16:10"; // bad form, do not use spaces
monitorInfo.colors = "16.7 million";
trace(monitorInfo["aspect ratio"], monitorInfo.colors);
// output: 16:10 16.7 million
```

La clé appelée `aspect ratio` contient un caractère d'espace. Cela est possible dans le cas de l'opérateur (`[]`) d'accès au tableau, mais avec l'opérateur point, une erreur se produit. L'utilisation d'espace dans le nom de vos clés n'est donc pas conseillée.

La seconde méthode pour créer un tableau associatif consiste à utiliser le constructeur Array (ou le constructeur d'une classe dynamique), puis à utiliser l'opérateur (`[]`) d'accès au tableau ou l'opérateur point (`.`) pour ajouter les paires de clés et de valeurs au tableau. Si vous déclarez votre tableau associatif comme étant de type Array, vous ne pouvez pas utiliser de littéral d'objet pour l'initialiser. Ce code crée un tableau associatif appelé `monitorInfo` à l'aide du constructeur Array, et ajoute les clés appelées `type` et `resolution`, ainsi que leurs valeurs :

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"], monitorInfo["resolution"]);
// output: Flat Panel 1600 x 1200
```

L'utilisation du constructeur Array pour créer un tableau associatif ne présente aucun avantage. Vous ne pouvez pas utiliser la propriété `Array.length` ou une méthode de la classe Array avec des tableaux associatifs, même si vous utilisez le constructeur Array ou le type de données Array. Il est préférable d'utiliser le constructeur Array pour créer des tableaux indexés.

Tableaux associatifs avec clés d'objet (dictionnaires)

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la classe Dictionary pour créer un tableau associatif qui utilise des objets pour les clés au lieu de chaînes. Ces tableaux sont parfois appelés dictionnaires, hachages ou mappages. Par exemple, supposez que vous avez une application qui détermine l'emplacement d'un objet Sprite selon son association avec un conteneur spécifique. Vous pouvez utiliser un objet Dictionary pour mapper chaque objet Sprite à un conteneur.

Le code suivant crée trois occurrences de la classe Sprite qui servent de clés pour l'objet Dictionary. La valeur `GroupA` ou `GroupB` est affectée à chaque clé. Les valeurs peuvent être de n'importe quel type de données, mais dans cet exemple, `GroupA` et `GroupB` sont des occurrences de la classe Object. Vous pouvez ensuite accéder à la valeur associée à chaque clé avec l'opérateur d'accès au tableau (`[]`), comme illustré dans le code suivant :

```
import flash.display.Sprite;
import flash.utils.Dictionary;

var groupMap:Dictionary = new Dictionary();

// objects to use as keys
var spr1:Sprite = new Sprite();
var spr2:Sprite = new Sprite();
var spr3:Sprite = new Sprite();

// objects to use as values
var groupA:Object = new Object();
var groupB:Object = new Object();

// Create new key-value pairs in dictionary.
groupMap[spr1] = groupA;
groupMap[spr2] = groupB;
groupMap[spr3] = groupB;

if (groupMap[spr1] == groupA)
{
    trace("spr1 is in groupA");
}
if (groupMap[spr2] == groupB)
{
    trace("spr2 is in groupB");
}
if (groupMap[spr3] == groupB)
{
    trace("spr3 is in groupB");
}
```

Itération avec des clés d'objet

Vous pouvez parcourir en boucle le contenu d'un objet Dictionary à l'aide d'une boucle `for..in` ou d'une boucle `for each..in`. Une boucle `for..in` vous permet d'effectuer une itération en fonction des clés, tandis qu'une boucle `for each..in` vous permet d'effectuer une itération en fonction des valeurs associées à chaque clé.

Utilisez la boucle `for..in` pour accéder directement aux clés d'objet d'un objet Dictionary. Vous pouvez également accéder aux valeurs de l'objet Dictionary avec l'opérateur d'accès au tableau (`[]`). Le code suivant utilise l'exemple précédent du dictionnaire `groupMap` pour indiquer comment parcourir en boucle un objet Dictionary avec la boucle `for..in` :

```
for (var key:Object in groupMap)
{
    trace(key, groupMap[key]);
}
/* output:
[object Sprite] [object Object]
[object Sprite] [object Object]
[object Sprite] [object Object]
*/
```

Utilisez la boucle `for each..in` pour accéder directement aux valeurs d'un objet Dictionary. Le code suivant utilise également le dictionnaire `groupMap` pour indiquer comment parcourir en boucle un objet Dictionary avec la boucle `for each..in` :

```
for each (var item:Object in groupMap)
{
    trace(item);
}
/* output:
[object Object]
[object Object]
[object Object]
*/
```

Clés d'objet et gestion de la mémoire

Adobe® Flash® Player et Adobe® AIR™ utilisent un système de nettoyage permettant de récupérer la mémoire qui n'est plus utilisée. Lorsque aucune référence ne pointe vers un objet, ce dernier peut être nettoyé et la mémoire récupérée au prochain nettoyage. Par exemple, le code suivant crée un objet et lui affecte une référence à la variable `myObject` :

```
var myObject:Object = new Object();
```

Tant que des références à l'objet existent, le système de nettoyage ne récupère pas la mémoire que l'objet occupe. Si la valeur de `myObject` est modifiée et qu'elle pointe vers un autre objet ou qu'elle est définie sur `null`, la mémoire occupée par l'objet d'origine peut être nettoyée. Néanmoins, aucune autre référence à l'objet d'origine ne doit exister.

Si vous utilisez `myObject` comme clé dans un objet Dictionary, vous créez une autre référence à l'objet d'origine. Par exemple, le code suivant crée deux références à un objet, la variable `myObject` et la clé dans l'objet `myMap` :

```
import flash.utils.Dictionary;

var myObject:Object = new Object();
var myMap:Dictionary = new Dictionary();
myMap[myObject] = "foo";
```

Pour que l'objet référencé par `myObject` puisse être nettoyé, vous devez supprimer toutes ses références. Dans ce cas, vous devez modifier la valeur de `myObject` et supprimer la clé `myObject` de `myMap`, comme indiqué dans le code suivant :

```
myObject = null;
delete myMap[myObject];
```

Vous pouvez également utiliser le paramètre `useWeakReference` du constructeur Dictionary pour que toutes les clés de dictionnaire deviennent des *références faibles*. Le système de nettoyage ignore les références faibles. Par conséquent, un objet n'ayant que des références faibles peut être nettoyé. Par exemple, dans le code suivant, vous n'avez pas besoin de supprimer la clé `myObject` de `myMap` pour que l'objet puisse être nettoyé :

```
import flash.utils.Dictionary;

var myObject:Object = new Object();
var myMap:Dictionary = new Dictionary(true);
myMap[myObject] = "foo";
myObject = null; // Make object eligible for garbage collection.
```

Tableaux multidimensionnels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les tableaux multidimensionnels contiennent d'autres tableaux comme éléments. Prenons par exemple une liste de tâches stockée sous forme de tableau indexé de chaînes :

```
var tasks:Array = ["wash dishes", "take out trash"];
```

Pour stocker une liste de tâches distincte pour chaque jour de la semaine, vous pouvez créer un tableau multidimensionnel avec un élément pour chaque jour. Chaque élément contient à son tour un tableau indexé (semblable au tableau `tasks`) qui stocke la liste des tâches. Vous pouvez utiliser n'importe quelle combinaison de tableaux indexés ou associatifs dans des tableaux multidimensionnels. Les exemples des sections suivantes utilisent soit deux tableaux indexés soit un tableau associatif de tableaux indexés. Vous pouvez essayer les autres combinaisons pour vous exercer.

Deux tableaux indexés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous utilisez deux tableaux indexés, vous pouvez visualiser le résultat sous forme de tableau ou de feuille de calcul. Les éléments du premier tableau représentent les lignes alors que les éléments du second tableau représentent les colonnes.

Par exemple, le tableau multidimensionnel suivant utilise deux tableaux indexés pour suivre des listes de tâches pour chaque jour de la semaine. Le premier tableau, `masterTaskList`, est créé à l'aide du constructeur de classe Array. Chaque élément du tableau représente un jour de la semaine, avec l'index 0 représentant lundi et l'index 6 dimanche. Ces éléments peuvent être considérés comme les lignes du tableau. Vous pouvez créer la liste de tâches de chaque jour en affectant un littéral de tableau à chacun des sept éléments que vous créez dans le tableau `masterTaskList`. Les littéraux de tableau représentent les colonnes du tableau.

```
var masterTaskList:Array = new Array();
masterTaskList[0] = ["wash dishes", "take out trash"];
masterTaskList[1] = ["wash dishes", "pay bills"];
masterTaskList[2] = ["wash dishes", "dentist", "wash dog"];
masterTaskList[3] = ["wash dishes"];
masterTaskList[4] = ["wash dishes", "clean house"];
masterTaskList[5] = ["wash dishes", "wash car", "pay rent"];
masterTaskList[6] = ["mow lawn", "fix chair"];
```

Vous pouvez accéder à des éléments particuliers sur toute liste des tâches à l'aide de l'opérateur d'accès au tableau (`[]`). Le premier groupe de crochets représente le jour de la semaine et le second la liste de tâches pour ce jour. Par exemple, pour récupérer la seconde tâche de la liste du mercredi, utilisez d'abord l'index 2 pour mercredi puis utilisez l'index 1 pour la seconde tâche dans la liste.

```
trace(masterTaskList[2][1]); // output: dentist
```

Pour récupérer la première tâche de la liste du dimanche, utilisez l'index 6 pour dimanche et l'index 0 pour la première tâche sur la liste.

```
trace(masterTaskList[6][0]); // output: mow lawn
```

Tableau associatif avec un tableau indexé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour faciliter l'accès aux tableaux, vous pouvez utiliser un tableau associatif pour les jours de la semaine et un tableau indexé pour les listes de tâche. Les tableaux associatifs vous permettent d'utiliser une syntaxe à point lorsque vous vous référez à un jour particulier de la semaine, mais nécessitent un traitement d'exécution supplémentaire pour accéder à chaque élément du tableau associatif. L'exemple suivant utilise un tableau associatif comme base d'une liste de tâches, avec une paire de clés et de valeurs pour chaque jour de la semaine :

```
var masterTaskList:Object = new Object();
masterTaskList["Monday"] = ["wash dishes", "take out trash"];
masterTaskList["Tuesday"] = ["wash dishes", "pay bills"];
masterTaskList["Wednesday"] = ["wash dishes", "dentist", "wash dog"];
masterTaskList["Thursday"] = ["wash dishes"];
masterTaskList["Friday"] = ["wash dishes", "clean house"];
masterTaskList["Saturday"] = ["wash dishes", "wash car", "pay rent"];
masterTaskList["Sunday"] = ["mow lawn", "fix chair"];
```

La syntaxe à point rend le code plus lisible car elle évite d'utiliser plusieurs groupes de crochets.

```
trace(masterTaskList.Wednesday[1]); // output: dentist
trace(masterTaskList.Sunday[0]);// output: mow lawn
```

Vous pouvez parcourir en boucle la liste des tâches en utilisant une boucle `for..in`, mais vous devez utiliser l'opérateur d'accès au tableau (`[]`), en lieu et place de la syntaxe à point, pour accéder à la valeur associée à chaque clé. Etant donné que `masterTaskList` est un tableau associatif, les éléments ne sont pas nécessairement récupérés dans l'ordre que vous attendez, comme l'indique l'exemple suivant :

```
for (var day:String in masterTaskList)
{
    trace(day + ": " + masterTaskList[day])
}
/* output:
Sunday: mow lawn,fix chair
Wednesday: wash dishes,dentist,wash dog
Friday: wash dishes,clean house
Thursday: wash dishes
Monday: wash dishes,take out trash
Saturday: wash dishes,wash car,pay rent
Tuesday: wash dishes,pay bills
*/
```

Clonage de tableaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Array ne possède aucune méthode intégrée pour effectuer des copies de tableaux. Vous pouvez créer une *copie simple* d'un tableau en appelant la méthode `concat()` ou `slice()` sans arguments. Dans une copie simple, si le tableau d'origine a des éléments qui sont des objets, seules les références aux objets sont copiées (et non les objets). La copie pointe vers les mêmes objets que l'original. Tout changement effectué sur les objets apparaît dans les deux tableaux.

Dans une *copie en profondeur*, les objets se trouvant dans le tableau d'origine sont copiés également de façon à ce que le nouveau tableau ne pointe pas vers les mêmes objets que le tableau d'origine. La copie en profondeur exige plus d'une ligne de code, ce qui nécessite généralement la création d'une fonction. Une telle fonction peut être créée comme fonction d'utilitaire globale ou comme méthode d'une sous-classe Array.

L'exemple suivant définit une fonction appelée `clone()` qui effectue une copie en profondeur. L'algorithme est issu d'une technique de programmation Java courante. La fonction crée une copie en profondeur en sérialisant le tableau en une occurrence de la classe ByteArray puis en relisant le tableau dans un nouveau tableau. Cette fonction accepte un objet de façon à ce qu'il puisse être utilisé à la fois avec des tableaux indexés et des tableaux associatifs, comme indiqué dans le code suivant :

```
import flash.utils.ByteArray;

function clone(source:Object):*
{
    var myBA:ByteArray = new ByteArray();
    myBA.writeObject(source);
    myBA.position = 0;
    return(myBA.readObject());
}
```

Extension de la classe Array

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Array est l'une des classes de base non finale, c'est-à-dire que vous pouvez créer votre sous-classe d'Array. Cette section décrit comment créer une sous-classe d'Array et décrit les problèmes pouvant se poser pendant le processus.

Comme mentionné précédemment, les tableaux dans ActionScript ne sont pas typés, mais vous pouvez créer une sous-classe d'Array qui accepte des éléments d'un seul type de données. L'exemple fourni dans les sections suivantes définit une sous-classe Array appelée TypedArray qui limite ses éléments à des valeurs du type de données indiqué dans le premier paramètre. La classe TypedArray est présentée comme un exemple de la façon dont la classe Array est étendue et risque de ne pas être adaptée à des fins de production pour différentes raisons. Premièrement, la vérification du type a lieu lors de l'exécution plutôt que de la compilation. Deuxièmement, lorsqu'une méthode TypedArray rencontre une incompatibilité, elle est ignorée et aucune exception n'est renvoyée, même si vous pouvez facilement modifier les méthodes pour renvoyer des exceptions. Troisièmement, la classe ne peut pas empêcher l'utilisation de l'opérateur d'accès au tableau pour insérer des valeurs de n'importe quel type dans le tableau. Quatrièmement, le style de codage privilégie la simplicité par rapport à l'optimisation des performances.

Remarque : *vous pouvez utiliser la technique décrite ici pour créer un tableau typé. Cependant, utiliser un objet Vector constitue une meilleure démarche. Une occurrence de Vector est un véritable tableau typé. Elle dépasse la classe Array ou toute sous-classe par ses performances et ses améliorations. Une illustration de la création d'une sous-classe Array constitue l'objet de cette description.*

Déclaration de la sous-classe

Utilisez le mot-clé `extends` pour indiquer qu'une classe est une sous-classe d'Array. Une sous-classe d'Array doit utiliser l'attribut `dynamic`, comme la classe Array. Autrement, votre sous-classe ne fonctionne pas correctement.

Le code suivant représente la définition de la classe TypedArray, qui comporte une constante contenant le type de données, une méthode de constructeur et les quatre méthodes permettant d'ajouter des éléments au tableau. Le code pour chaque méthode est omis dans cet exemple, mais il est décrit de façon détaillée dans les sections qui suivent :

```
public dynamic class TypedArray extends Array
{
    private const dataType:Class;

    public function TypedArray(...args) {}

    AS3 override function concat(...args):Array {}

    AS3 override function push(...args):uint {}

    AS3 override function splice(...args) {}

    AS3 override function unshift(...args):uint {}
}
```

Les quatre méthodes de remplacement utilisent l'espace de nom AS3 au lieu de l'attribut `public` car cet exemple suppose que l'option `-as3` du compilateur est définie sur `true` et l'option `-es` du compilateur sur `false`. Il s'agit des paramètres par défaut pour Adobe Flash Builder et AdobeFlashProfessional.

Si vous êtes un développeur expérimenté et que vous préférez utiliser l'héritage de prototype, vous pouvez apporter deux changements mineurs à la classe TypedArray afin qu'elle compile avec l'option `-es` du compilateur définie sur `true`. Commencez par supprimer toutes les occurrences de l'attribut `override` et remplacez l'espace de nom AS3 par l'attribut `public`. Remplacez ensuite `Array.prototype` pour les quatre occurrences de `super`.

Constructeur TypedArray

Le constructeur de sous-classe pose un défi intéressant car il doit accepter une liste d'arguments de longueur arbitraire. Il s'agit de savoir comment transférer les arguments au superconstructeur pour créer le tableau. Si vous transmettez la liste des arguments sous forme d'un tableau, le superconstructeur le considère comme un seul argument de type Array et le tableau résultant a toujours une longueur d'1 élément. Le transfert de listes d'arguments se fait généralement au moyen de la méthode `Function.apply()`, qui prend un tableau d'arguments comme second paramètre mais le convertit en une liste d'arguments lors de l'exécution de la fonction. Malheureusement, vous ne pouvez pas utiliser la méthode `Function.apply()` avec des constructeurs.

La seule solution est de recréer la logique du constructeur Array dans le constructeur TypedArray. Le code suivant indique l'algorithme utilisé dans le constructeur de classe Array, que vous pouvez réutiliser dans votre constructeur de sous-classe Array :

```
public dynamic class Array
{
    public function Array(...args)
    {
        var n:uint = args.length
        if (n == 1 && (args[0] is Number))
        {
            var dlen:Number = args[0];
            var ulen:uint = dlen;
            if (ulen != dlen)
            {
                throw new RangeError("Array index is not a 32-bit unsigned integer ("+dlen+")");
            }
            length = ulen;
        }
        else
        {
            length = n;
            for (var i:int=0; i < n; i++)
            {
                this[i] = args[i]
            }
        }
    }
}
```

Le constructeur TypedArray partage une grande partie du code du constructeur Array, avec seulement quatre changements apportés au code. Premièrement, la liste des paramètres comprend un nouveau paramètre obligatoire de type Class qui permet d'indiquer le type de données du tableau. Deuxièmement, le type de données transmis au constructeur est affecté à la variable `dataType`. Troisièmement, dans l'instruction `else`, la valeur de la propriété `length` est affectée après la boucle `for` de façon à ce que `length` comprenne uniquement des arguments du type correct. Quatrièmement, le corps de la boucle `for` utilise la version de remplacement de la méthode `push()` de façon à ce que seuls des arguments du type de données correct soient ajoutés au tableau. L'exemple suivant présente la fonction constructeur TypedArray :

```
public dynamic class TypedArray extends Array
{
    private var dataType:Class;
    public function TypedArray(typeParam:Class, ...args)
    {
        dataType = typeParam;
        var n:uint = args.length
        if (n == 1 && (args[0] is Number))
        {
            var dlen:Number = args[0];
            var ulen:uint = dlen
            if (ulen != dlen)
            {
                throw new RangeError("Array index is not a 32-bit unsigned integer ("+dlen+")")
            }
            length = ulen;
        }
        else
        {
            for (var i:int=0; i < n; i++)
            {
                // type check done in push()
                this.push(args[i])
            }
            length = this.length;
        }
    }
}
```

Méthodes de remplacement TypedArray

La classe TypedArray remplace les quatre méthodes de la classe Array qui permettent d'ajouter des éléments à un tableau. Dans chaque cas, la méthode de remplacement ajoute une vérification du type qui empêche d'ajouter des éléments qui ne sont pas du type de données correct. Chaque méthode appelle ensuite sa version de superclasse.

La méthode `push()` parcourt en boucle la liste des arguments avec une boucle `for..in` et effectue une vérification du type sur chaque argument. Les arguments qui ne sont pas de type correct sont supprimés du tableau `args` avec la méthode `splice()`. Une fois que la boucle `for..in` se termine, le tableau `args` contient des valeurs de type `dataType` uniquement. La version de superclasse de `push()` est ensuite appelée avec le tableau `args` mis à jour, comme indiqué dans le code suivant :

```
    AS3 override function push(...args):uint
    {
        for (var i:* in args)
        {
            if (!(args[i] is dataType))
            {
                args.splice(i,1);
            }
        }
        return (super.push.apply(this, args));
    }
```

La méthode `concat()` crée un TypedArray temporaire appelé `passArgs` pour stocker les arguments soumis à la vérification de type. Ceci permet de réutiliser le code de vérification de type qui existe dans la méthode `push()`. Une boucle `for..in` effectue une itération sur le tableau `args` et appelle `push()` sur chaque argument. Etant donné que `passArgs` est typé sous la forme TypedArray, la version TypedArray de `push()` est exécutée. La méthode `concat()` appelle ensuite sa version de superclasse, comme indiqué dans le code suivant :

```
AS3 override function concat(...args):Array
{
    var passArgs:TypedArray = new TypedArray(dataType);
    for (var i:* in args)
    {
        // type check done in push()
        passArgs.push(args[i]);
    }
    return (super.concat.apply(this, passArgs));
}
```

La méthode `splice()` prend une liste d'arguments arbitraire, mais les deux premiers arguments se réfèrent toujours à un numéro d'index et au nombre d'éléments à supprimer. C'est pourquoi la méthode de remplacement `splice()` effectue la vérification de type uniquement pour les éléments du tableau `args` dans les positions d'index 2 ou supérieures. Il est intéressant de noter que dans le code, il semble y avoir un appel récursif à `splice()` à l'intérieur de la boucle `for`, mais en réalité, ce n'est pas le cas car `args` est de type Array et non de type TypedArray, ce qui signifie que l'appel à `args.splice()` est un appel à la version de superclasse de la méthode. Une fois que la boucle `for..in` se termine, le tableau `args` contient des valeurs du type correct uniquement dans les positions d'index 2 ou supérieures, et `splice()` appelle sa version de superclasse, comme indiqué dans le code suivant :

```
AS3 override function splice(...args):*
{
    if (args.length > 2)
    {
        for (var i:int=2; i< args.length; i++)
        {
            if (!(args[i] is dataType))
            {
                args.splice(i,1);
            }
        }
    }
    return (super.splice.apply(this, args));
}
```

La méthode `unshift()`, qui ajoute des éléments au début d'un tableau, accepte une liste d'arguments arbitraire également. La méthode de remplacement `unshift()` utilise un algorithme très semblable à celui utilisé par la méthode `push()`, comme indiqué dans le code suivant :

```
AS3 override function unshift(...args):uint
{
    for (var i:* in args)
    {
        if (!(args[i] is dataType))
        {
            args.splice(i,1);
        }
    }
    return (super.unshift.apply(this, args));
}
}
```

Exemple de tableau : PlayList

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple PlayList présente les techniques d'utilisation des tableaux, dans le contexte d'une application de lecture musicale qui gère une liste de chansons. Ces techniques sont les suivantes :

- Création d'un tableau indexé
- Ajout d'éléments à un tableau indexé
- Tri d'un tableau d'objets en fonction de différentes propriétés, à l'aide d'options de tri différentes
- Conversion d'un tableau en une chaîne séparée par des caractères

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application PlayList se trouvent dans le dossier Samples/PlayList. L'application se compose des fichiers suivants :

Fichier	Description
PlayList.mxml ou PlayList.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/playlist/PlayList.as	Classe représentant une liste de morceaux. Elle utilise un tableau pour enregistrer la liste et gère le tri des éléments de la liste.
com/example/programmingas3/playlist/Song.as	Objet de valeur représentant des informations sur une seule chanson. Les éléments gérés par la classe PlayList sont des occurrences Song.
com/example/programmingas3/playlist/SortProperty.as	Pseudo-énumération dont les valeurs disponibles représentent les propriétés de la classe Song en fonction desquelles une liste d'objets Song peut être triée.

Présentation de la classe PlayList

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe PlayList gère un ensemble d'objets Song. Elle a des méthodes publiques qui permettent d'ajouter une chanson à la liste de lecture (la méthode `addSong()`) et de trier les chansons dans la liste (la méthode `sortList()`). En outre, la classe comprend une propriété d'accesseur en lecture seule, `songList`, qui permet d'accéder au groupe de chansons dans la liste de lecture. En interne, la classe PlayList conserve une trace de ses chansons à l'aide d'une variable Array privée :

```
public class PlayList
{
    private var _songs:Array;
    private var _currentSort:SortProperty = null;
    private var _needToSort:Boolean = false;
    ...
}
```

En plus de la variable Array `_songs` utilisée par la classe PlayList pour conserver une trace de sa liste de chansons, deux autres variables privées vérifient si la liste doit être triée (`_needToSort`) et contrôlent la propriété sur laquelle est basé le tri de la liste de chansons à un moment donné (`_currentSort`).

Comme avec tous les objets, lorsque vous avez déclaré une occurrence de Array, vous n'avez effectué que la moitié du travail consistant à créer un tableau. Avant d'accéder à des méthodes ou à des propriétés d'une occurrence de Array, cette dernière doit être instanciée dans le constructeur de la classe PlayList.

```
public function PlayList()
{
    this._songs = new Array();
    // Set the initial sorting.
    this.sortList(SortProperty.TITLE);
}
```

La première ligne du constructeur instancie la variable `_songs` pour qu'elle puisse être utilisée. En outre, la méthode `sortList()` est appelée pour définir la propriété de tri initiale.

Ajout d'une chanson à la liste

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un utilisateur ajoute une nouvelle chanson dans l'application, le code dans le formulaire de saisie des données appelle la méthode `addSong()` de la classe PlayList.

```
/**
 * Adds a song to the playlist.
 */
public function addSong(song:Song):void
{
    this._songs.push(song);
    this._needToSort = true;
}
```

A l'intérieur de `addSong()`, la méthode `push()` du tableau `_songs` est appelée. Ceci permet d'ajouter l'objet Song transmis à `addSong()` en tant que nouvel élément dans ce tableau. Avec la méthode `push()` , le nouvel élément est ajouté à la fin du tableau, indépendamment du tri appliqué précédemment. Ceci signifie qu'une fois que la méthode `push()` a été appelée, la liste des chansons risque de ne plus être triée correctement. Par conséquent, la variable `_needToSort` est définie sur `true`. Théoriquement, la méthode `sortList()` pourrait être appelée immédiatement afin d'éviter de vérifier si la liste est triée ou non à un moment donné. En pratique, cependant, la liste des chansons n'a pas besoin d'être triée jusqu'au moment précédant immédiatement sa récupération. En retardant l'opération de tri, l'application n'effectue pas de tri inutile si, par exemple, plusieurs chansons sont ajoutées à la liste avant sa récupération.

Tri de la liste de chansons

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Etant donné que les occurrences Song gérées par la liste de lecture sont des objets complexes, les utilisateurs de l'application peuvent trier la liste de lecture en fonction de différentes propriétés (titre de la chanson ou année de publication, par exemple). Dans l'application PlayList, le tri de la liste des chansons s'effectue en trois étapes : identification de la propriété sur laquelle est basé le tri de la liste, indication des options de tri à utiliser lors du tri en fonction de cette propriété et exécution du tri.

Propriétés de tri

Un objet Song conserve la trace de plusieurs propriétés, notamment le titre de la chanson, l'artiste, l'année de publication, le nom du fichier et un ensemble de genres sélectionné par l'utilisateur auquel la chanson appartient. Seules les trois premières propriétés sont pratiques pour le tri. Dans un souci de commodité pour les développeurs, l'exemple inclut la classe SortProperty, qui agit comme une énumération avec des valeurs représentant les propriétés disponibles pour le tri.

```
public static const TITLE:SortProperty = new SortProperty("title");
public static const ARTIST:SortProperty = new SortProperty("artist");
public static const YEAR:SortProperty = new SortProperty("year");
```

La classe SortProperty contient trois classes, `TITLE`, `ARTIST` et `YEAR`. Chacune d'elles stocke une chaîne comportant le nom de la propriété de la classe Song pouvant être utilisée pour le tri. Chaque fois qu'une propriété de tri est indiquée dans le reste du code, le membre de l'énumération est utilisé. Par exemple, dans le constructeur PlayList, la liste est triée initialement en appelant la méthode `sortList()`, comme suit :

```
// Set the initial sorting.
this.sortList(SortProperty.TITLE);
```

Etant donné que la propriété de tri est spécifiée sous la forme `SortProperty.TITLE`, les chansons sont triées par titre.

Tri par propriété et définition d'options de tri

La classe PlayList trie la liste de chansons dans la méthode `sortList()`, comme suit :

```
/**
 * Sorts the list of songs according to the specified property.
 */
public function sortList(sortProperty:SortProperty):void
{
    ...
    var sortOptions:uint;
    switch (sortProperty)
    {
        case SortProperty.TITLE:
            sortOptions = Array.CASEINSENSITIVE;
            break;
        case SortProperty.ARTIST:
            sortOptions = Array.CASEINSENSITIVE;
            break;
        case SortProperty.YEAR:
            sortOptions = Array.NUMERIC;
            break;
    }

    // Perform the actual sorting of the data.
    this._songs.sortOn(sortProperty.propertyName, sortOptions);

    // Save the current sort property.
    this._currentSort = sortProperty;

    // Record that the list is sorted.
    this._needToSort = false;
}
```

Lors du tri par titre ou par artiste, il est préférable d'effectuer un tri par ordre alphabétique. En revanche, lors du tri par année, il est plus logique d'effectuer un tri numérique. L'instruction `switch` sert à définir l'option de tri appropriée, stockée dans la variable `sortOptions`, en fonction de la valeur indiquée dans le paramètre `sortProperty`. Ici encore, les membres de l'énumération nommés sont utilisés pour faire la différence entre les propriétés, plutôt que les valeurs absolues.

Une fois que vous avez déterminé la propriété et les options de tri, le tableau `_songs` est trié en appelant sa méthode `sortOn()`, en transmettant ces deux valeurs comme paramètres. La propriété de tri est enregistrée et la liste des chansons est triée.

Combinaison d'éléments de tableau en une chaîne séparée par des caractères

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple utilise non seulement un tableau pour conserver la liste des chansons dans la classe PlayList mais également des tableaux dans la classe Song pour gérer la liste des genres auxquels une chanson appartient. Considérons ce fragment de code issu de la définition de la classe Song :

```
private var _genres:String;

public function Song(title:String, artist:String, year:uint, filename:String, genres:Array)
{
    ...
    // Genres are passed in as an array
    // but stored as a semicolon-separated string.
    this._genres = genres.join(";");
}
```

Lors de la création d'une occurrence de Song, le paramètre `genres` utilisé pour spécifier le genre (ou les genres) auquel la chanson appartient est défini comme occurrence d'Array. Ainsi, vous pouvez regrouper plusieurs genres en une seule variable qui peut être transmise au constructeur. Néanmoins, la classe Song conserve, en interne, les genres dans la variable privée `_genres` sous la forme d'une occurrence de String séparée par des points-virgules. Le paramètre Array est converti en une chaîne séparée par des points-virgules en appelant sa méthode `join()` avec la valeur de chaîne littérale `";"` comme séparateur spécifié.

De la même façon, les accesseurs `genres` permettent de définir ou de récupérer des genres sous la forme d'un tableau :

```
    public function get genres():Array
    {
        // Genres are stored as a semicolon-separated String,
        // so they need to be transformed into an Array to pass them back out.
        return this._genres.split(";");
    }
    public function set genres(value:Array):void
    {
        // Genres are passed in as an array,
        // but stored as a semicolon-separated string.
        this._genres = value.join(";");
    }
```

L'accesseur `genresset` se comporte exactement comme le constructeur ; il accepte un tableau et appelle la méthode `join()` pour la convertir en une chaîne séparée par des points-virgules. L'accesseur `get` effectue l'opération inverse : la méthode `split()` de la variable `_genres` est appelée. Elle divise la chaîne en un tableau de valeurs utilisant le séparateur spécifié (la valeur de chaîne littérale `";"` comme précédemment).

Chapitre 4 : Gestion des erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

« Gérer » une erreur signifie que vous intégrez des fonctions logiques à l'application pour réagir à une erreur ou la corriger. Les erreurs sont générées lors de la compilation d'une application ou lors de l'exécution d'une application compilée. Lorsque l'application gère les erreurs, il se produit une *réaction* à l'erreur. Il arrive en revanche qu'une erreur soit ignorée (auquel cas le processus à l'origine de l'erreur échoue silencieusement). La gestion des erreurs, lorsqu'elle est utilisée correctement, protège votre application et ses utilisateurs contre un comportement inattendu.

Cependant, la gestion des erreurs est une catégorie large qui englobe la réponse à de nombreux types d'erreurs générées lors de la phase de compilation ou lors de l'exécution d'une application. Nous allons passer en revue la gestion des erreurs d'exécution (renvoyées lors de l'exécution d'une application), les différents types d'erreurs générés et les avantages du système de gestion des erreurs d'ActionScript 3.0.

Principes de base de la gestion des erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une erreur d'exécution est une erreur qui se produit dans votre code ActionScript et qui empêche le contenu ActionScript de s'exécuter comme prévu. Pour assurer l'exécution correcte du code ActionScript du point de vue de l'utilisateur, écrivez le code dans l'application qui gère l'erreur (la corrige, la contourne ou informe au moins l'utilisateur qu'elle a eu lieu). Ce processus est appelé *gestion des erreurs*.

La gestion des erreurs est une catégorie large qui englobe la réponse à de nombreux types d'erreurs générées lors de la phase de compilation ou lors de l'exécution d'une application. Les erreurs qui se produisent lors de la compilation sont souvent plus faciles à identifier. Corrigez-les pour terminer la création d'un fichier SWF.

Les erreurs d'exécution peuvent être difficiles à détecter car elles se produisent lorsque le code erroné est exécuté. Si un segment de votre programme contient plusieurs branches de code, telle une instruction `if..then..else`, testez toutes les conditions possibles, avec toutes les valeurs en entrée susceptibles d'être utilisées par un utilisateur réel, pour confirmer que le code ne contient pas d'erreur.

Les erreurs d'exécution peuvent être divisées en deux catégories : les *erreurs de programme* sont des erreurs dans votre code ActionScript (spécification du type de données incorrect pour un paramètre de méthode, par exemple) ; les *erreurs logiques* sont des erreurs dans la logique (le contrôle des données et la manipulation des valeurs) de votre programme (utilisation de la formule incorrecte pour calculer les taux d'intérêt dans une application bancaire, par exemple). Encore une fois, ces deux types d'erreurs peuvent souvent être détectés et corrigés à l'avance en testant attentivement votre application.

Il serait idéal d'identifier et de supprimer toutes les erreurs de votre application avant de la mettre à la disposition des utilisateurs finaux. Cependant, toutes les erreurs ne peuvent pas être prévues ni évitées. Supposons, par exemple, que l'application ActionScript charge des informations depuis un site Web particulier sur lequel vous n'avez aucun contrôle. Si ce site Web n'est pas disponible, la partie de l'application qui dépend de ces données externes ne se comporte pas correctement. L'aspect primordial de la gestion des erreurs consiste à anticiper ces cas de figure et à les traiter judicieusement. Il est préférable que l'exécution de l'application ne soit pas interrompue ou, tout du moins, qu'un message indique à l'utilisateur pourquoi elle ne fonctionne pas.

Les erreurs d'exécution sont représentées de deux façons dans ActionScript :

- Classes d'erreur : de nombreuses erreurs sont associées à une classe Error. Lorsqu'une erreur se produit, le moteur d'exécution Flash (Flash Player ou Adobe AIR, par exemple) crée une occurrence de la classe Error spécifique associée à cette erreur. Votre code peut utiliser les informations contenues dans cet objet erreur pour donner une réponse appropriée à l'erreur.
- Evénements d'erreur : il arrive qu'une erreur se produise lorsque le moteur d'exécution Flash déclencherait normalement un événement. Si tel est le cas, un événement d'erreur est alors déclenché. Chaque événement d'erreur étant associé à une classe, le moteur d'exécution de Flash transmet une occurrence de cette classe aux méthodes enregistrées auprès de l'événement d'erreur.

Pour déterminer si une méthode donnée peut déclencher une erreur ou un événement d'erreur, voir la rubrique correspondante dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la programmation de routines de gestion des erreurs :

Asynchrone Commande de programme telle qu'un appel de méthode qui ne fournit pas un résultat immédiat, mais qui produit un résultat (ou une erreur) sous la forme d'un événement.

Capture Lorsqu'une exception (une erreur d'exécution) se produit et que votre code la découvre, ce dernier la *capture*. Lorsqu'une exception est capturée, le moteur d'exécution Flash cesse d'indiquer à un autre code ActionScript que l'exception s'est produite.

Version de débogage Version spéciale du moteur d'exécution Flash, telle que la version de débogage de Flash Player ou l'application de débogage du lanceur AIR (ADL), qui contient le code requis pour avertir les utilisateurs de la présence d'erreurs d'exécution. Dans la version standard de Flash Player ou Adobe AIR (celle que possèdent la plupart des utilisateurs), les erreurs qui ne sont pas gérées par votre code ActionScript sont ignorées. Dans les versions de débogage (intégrées à Adobe Flash CS4 Professional et Adobe Flash Builder), un message d'avertissement apparaît lorsqu'une erreur non gérée se produit.

Exception Erreur qui se produit lorsqu'une application est en cours d'exécution et que le moteur d'exécution Flash ne peut pas la résoudre seul.

Renvoi Lorsque votre code capture une exception, le moteur d'exécution Flash cesse de signaler l'exception à d'autres objets. S'il est important pour d'autres objets que l'exception leur soit signalée, le code doit *renvoyer* l'exception pour recommencer le processus de notification.

Synchrone Commande de programme (un appel de méthode, par exemple) qui fournit un résultat immédiat (ou qui renvoie immédiatement une erreur), ce qui signifie que la réponse peut être utilisée dans le même bloc de code.

Envoi Le fait de signaler au moteur d'exécution Flash (et par conséquent, à d'autres objets et au code ActionScript) qu'une erreur s'est produite s'appelle *envoyer* une erreur.

Types d'erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous développez et exécutez des applications, vous rencontrez différents types d'erreurs et de termes. La liste suivante présente les principaux termes et types d'erreurs :

- *Erreurs de compilation* : générées par le compilateur ActionScript lors de la compilation du code. Les erreurs de compilation ont lieu lorsque des problèmes de syntaxe dans votre code empêchent de créer votre application.
- *Erreurs d'exécution* : générées lorsque vous exécutez votre application après l'avoir compilée. Les erreurs d'exécution représentent des erreurs qui se produisent lors de la lecture d'un fichier SWF dans un moteur d'exécution Flash tel qu'Adobe Flash Player ou Adobe AIR. Dans la plupart des cas, il est possible de gérer les erreurs d'exécution au moment où elles se produisent, de les signaler à l'utilisateur et de prendre les mesures requises pour poursuivre l'exécution de l'application. S'il s'agit d'une erreur grave (impossibilité de se connecter à un site Web distant ou de charger des données), vous pouvez utiliser la gestion des erreurs pour mettre fin à l'application en douceur.
- *Erreurs synchrones* : erreurs d'exécution générées lorsqu'une fonction est appelée. Par exemple, lorsque vous tentez d'utiliser une méthode spécifique et que l'argument que vous lui transmettez n'est pas valide, le moteur d'exécution de Flash renvoie une exception. La plupart des erreurs se produisent en mode synchrone (au moment de l'exécution d'une instruction) et le flux de contrôle passe immédiatement à l'instruction `catch` la plus appropriée.

  Par exemple, l'extrait de code suivant renvoie une erreur d'exécution, car la méthode `browse()` n'est pas appelée avant que le programme ne tente de charger un fichier :

  ```
  var fileRef:FileReference = new FileReference();
  try
  {
      fileRef.upload(new URLRequest("http://www.yourdomain.com/fileupload.cfm"));
  }
  catch (error:IllegalOperationError)
  {
      trace(error);
      // Error #2037: Functions called in incorrect sequence, or earlier
      // call was unsuccessful.
  }
  ```

  Dans ce cas, une erreur d'exécution est renvoyée de façon synchrone car Flash Player a déterminé que la méthode `browse()` n'a pas été appelée avant la tentative de chargement du fichier.

  Pour obtenir des informations détaillées relatives à la gestion des erreurs synchrones, voir « Gestion des erreurs synchrones dans une application » à la page 59.

- Les *erreursasynchrones* sont des erreurs du moteur d'exécution qui se produisent hors du flux normal du programme. Elles génèrent des événements, interceptés par des écouteurs d'événement. Une opération asynchrone est une opération dans laquelle une fonction lance une opération mais n'attend pas qu'elle se termine. Vous pouvez créer un écouteur d'événements d'erreur pour attendre que l'application ou l'utilisateur tente une opération. Si cette dernière échoue, vous interceptez l'erreur avec un écouteur d'événements et répondez à l'événement d'erreur. Ensuite, l'écouteur d'événement appelle une fonction de gestionnaire d'événement pour répondre à l'événement d'erreur avec pertinence. Par exemple, le gestionnaire d'événement peut lancer une boîte de dialogue qui invite l'utilisateur à résoudre l'erreur.

Reprenez l'exemple d'erreur synchrone lors du chargement d'un fichier présenté précédemment. Si vous réussissez à appeler la méthode `browse()` avant de lancer le chargement d'un fichier, Flash Player distribue plusieurs événements. Par exemple, au démarrage d'un chargement, l'événement `open` est distribué. A la fin du chargement, l'événement `complete` est distribué. Etant donné que la gestion d'événements est asynchrone (c'est-à-dire qu'elle n'a pas lieu à des moments prédéfinis, connus et spécifiques), faites appel à la méthode `addEventListener()` pour détecter ces événements spécifiques, comme l'indique le code suivant :

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.OPEN, openHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
fileRef.browse();

function selectHandler(event:Event):void
{
    trace("...select...");
    var request:URLRequest = new URLRequest("http://www.yourdomain.com/fileupload.cfm");
    request.method = URLRequestMethod.POST;
    event.target.upload(request);
}
function openHandler(event:Event):void
{
    trace("...open...");
}
function completeHandler(event:Event):void
{
    trace("...complete...");
}
```

Pour obtenir des informations détaillées sur la gestion des erreurs asynchrones, voir « Réponse à des événements et à l'état d'erreur » à la page 65.

- *Exceptions non interceptées* : renvoyées sans logique correspondante (telle une instruction `catch`) pour y répondre. Si votre application renvoie une erreur, et qu'aucune instruction `catch` ni gestionnaire d'événement approprié n'est trouvé au niveau actuel ou supérieur pour gérer l'erreur, cette dernière est considérée comme une exception non interceptée.

  Lorsqu'il se produit une erreur non interceptée, le moteur d'exécution distribue un événement `uncaughtError`. Cet événement porte également le nom de « gestionnaire d'erreur global ». Il est distribué par l'objet UncaughtErrorEvents du fichier SWF et est proposé par la propriété `LoaderInfo.uncaughtErrorEvents`. Si aucun écouteur n'est enregistré pour l'événement `uncaughtError`, le moteur d'exécution ignore les erreurs non interceptées et tente de poursuivre son exécution, dès lors que l'erreur n'interrompt pas le fichier SWF.

  Outre la distribution de l'événement `uncaughtError`, les versions de débogage du moteur d'exécution de Flash répondent aux erreurs non interceptées en mettant fin au script actif. Elles affichent ensuite les erreurs non interceptées dans le résultat de l'instruction `trace` ou écrivent le message d'erreur dans un fichier journal. Si l'objet exception est une occurrence de la classe Error ou de l'une de ses sous-classes, les informations de trace de la pile s'affichent également dans le résultat. Pour plus d'informations sur l'utilisation de la version de débogage des moteurs d'exécution Flash, voir « Utilisation des versions de débogage des moteurs d'exécution Flash » à la page 58.

  Remarque : *lors du traitement d'un événement uncaughtError, si un événement d'erreur est renvoyé par un gestionnaire uncaughtError, celui-ci est appelé plusieurs fois. Il se produit alors une boucle infinie d'exceptions. Il est donc recommandé d'éviter ce type de scénario.*

Gestion des erreurs dans ActionScript 3.0

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Etant donné que de nombreuses applications peuvent être exécutées sans créer de logique pour gérer les erreurs, les développeurs sont tentés de retarder la création de la gestion des erreurs dans leurs applications. Néanmoins, sans gestion des erreurs, une application risque de s'interrompre ou de poser des problèmes à l'utilisateur si elle ne fonctionne pas comme prévu. ActionScript 2.0 possède une classe Error qui vous permet de créer une logique dans des fonctions personnalisées afin de renvoyer une exception avec un message spécifique. Etant donné que la gestion des erreurs est cruciale pour rendre une application conviviale, ActionScript 3.0 inclut une architecture étendue pour intercepter les erreurs.

Remarque : *bien que le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash passe en revue les exceptions renvoyées par de nombreuses méthodes, il ne contient pas nécessairement toutes les exceptions associées à chaque méthode. une méthode risque de renvoyer une exception due à une erreur de syntaxe ou d'autres problèmes qui ne sont pas signalés explicitement dans la description de la méthode, même si cette dernière répertorie certaines exceptions renvoyées.*

Eléments de gestion des erreurs ActionScript 3.0

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 comprend de nombreux outils permettant de gérer les erreurs, notamment :

- Classes Error : ActionScript 3.0 comprend un large éventail de classes Error destinées à multiplier le nombre de situations susceptibles de produire des objets d'erreur. Chaque classe Error permet aux applications de gérer et de répondre à des conditions d'erreur spécifiques, qu'elles soient liées à des erreurs système (comme une condition MemoryError), à des erreurs de codage (comme une condition ArgumentError), à des erreurs de réseau et de communication (comme une condition URIError), ou d'autres situations. Pour plus d'informations sur chaque classe, voir « Comparaison des classes Error » à la page 68.
- Moins d'échecs silencieux : dans les versions précédentes de Flash Player, les erreurs étaient générées et signalées uniquement si vous utilisiez explicitement l'instruction `throw`. Les méthodes et propriétés ActionScript natives renvoient des erreurs d'exécution pour le moteur d'exécution de Flash Player 9 et des versions ultérieures de Flash. Ces erreurs permettent de gérer les exceptions de manière plus efficace au moment où elles se produisent, puis de réagir à chaque exception.
- Messages d'erreur clairs affichés lors du débogage : Lorsque vous utilisez la version de débogage d'un moteur d'exécution de Flash, les situations ou le code à l'origine du problème génèrent des messages d'erreur détaillés qui vous aident à identifier les raisons de l'échec d'un bloc de code particulier. Ces messages optimisent la résolution des erreurs. Pour plus d'informations, voir « Utilisation des versions de débogage des moteurs d'exécution Flash » à la page 58.
- Les erreurs précises permettent d'afficher des messages d'erreur clairs pour les utilisateurs. Dans les versions précédentes de Flash Player, la méthode `FileReference.upload()` renvoyait la valeur booléenne `false` en cas d'échec de l'appel `upload()`, indiquant l'une des cinq erreurs possibles. Si une erreur se produit lorsque vous appelez la méthode `upload()` dans ActionScript 3.0, quatre erreurs spécifiques vous aident à afficher des messages d'erreur plus précis à l'intention des utilisateurs finaux.

- Gestion des erreurs affinée : des erreurs distinctes sont renvoyées pour de nombreuses situations courantes. Par exemple, dans ActionScript 2.0, avant qu'un objet FileReference ne soit renseigné, la propriété `name` possède la valeur `null` (par conséquent, avant d'utiliser ou d'afficher la propriété `name`, vérifiez qu'elle est définie sur une valeur autre que `null`). Dans ActionScript 3.0, si vous tentez d'accéder à la propriété `name` avant qu'elle ne soit renseignée, Flash Player ou AIR renvoie une erreur IllegalOperationError qui vous indique que la valeur n'a pas été définie. Vous pouvez utiliser des blocs `try..catch..finally` pour gérer l'erreur. Pour plus d'informations, voir « Utilisation des instructions try..catch..finally » à la page 59.
- Aucun problème sérieux de performance : l'utilisation de blocs `try..catch..finally` pour gérer des erreurs ne nécessite pas ou peu de ressources supplémentaires par rapport aux versions précédentes d'ActionScript.
- Une classe ErrorEvent qui vous permet de créer des écouteurs pour des événements d'erreurs asynchrones spécifiques : pour plus d'informations, voir « Réponse à des événements et à l'état d'erreur » à la page 65.

Stratégies de gestion des erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tant que l'application ne rencontre pas de condition problématique, vous pouvez continuer à l'exécuter sans créer de logique de gestion des erreurs dans le code. En revanche, si vous ne gérez pas d'erreurs de façon active et que votre application rencontre un problème, vos utilisateurs ignoreront toujours la raison de son échec.

Vous pouvez aborder la gestion des erreurs de diverses façons dans votre application. La liste suivante résume les trois principales options de gestion des erreurs :

- Utilisez les instructions `try..catch..finally`. Ces instructions interceptent les erreurs synchrones lorsqu'elles se produisent. Vous pouvez imbriquer vos instructions dans une hiérarchie pour intercepter des exceptions à différents niveaux d'exécution du code. Pour plus d'informations, voir « Utilisation des instructions try..catch..finally » à la page 59.
- Créez des objets d'erreur personnalisés. Vous pouvez utiliser la classe Error pour créer des objets d'erreur personnalisés afin de suivre des opérations spécifiques dans votre application qui ne sont pas couvertes par des types d'erreur intégrés. Vous pouvez ensuite appliquer des instructions `try..catch..finally` aux objets d'erreur personnalisés. Pour plus d'informations, voir « Création de classes d'erreur personnalisées » à la page 64.
- Ecrivez des gestionnaires et des écouteurs d'événement pour répondre à des événements d'erreur. Cette stratégie permet de créer des gestionnaires d'erreurs globaux destinés à gérer des événements similaires sans dupliquer un volume élevé de code dans les blocs `try..catch..finally`. Il est également plus probable que vous interceptiez des erreurs asynchrones à l'aide de cette approche. Pour plus d'informations, voir « Réponse à des événements et à l'état d'erreur » à la page 65.

Utilisation des versions de débogage des moteurs d'exécution Flash

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Adobe propose aux développeurs des éditions spéciales des moteurs d'exécution Flash, destinées à les aider à exécuter des opérations de débogage. Vous obtenez une copie de la version de débogage de Flash Player lorsque vous installez Adobe Flash Professional ou Adobe Flash Builder. Vous disposez également d'un utilitaire de débogage des applications Adobe AIR, appelé ADL, lorsque vous installez l'un de ces outils ou dans le cadre de l'installation du SDK d'Adobe AIR.

Il existe une grande différence dans la façon dont les versions de débogage et les versions de Flash Player et Adobe AIR mises sur le marché signalent les erreurs. Les versions de débogage indiquent le type d'erreur (Error, IOError ou EOFError générique), le numéro de l'erreur et un message d'erreur sous une forme lisible par une personne. Les versions mises sur le marché indiquent uniquement le type d'erreur et son numéro. Considérons par exemple le code qui suit :

```
try
{
    tf.text = myByteArray.readBoolean();
}
catch (error:EOFError)
{
    tf.text = error.toString();
}
```

Si la méthode `readBoolean()` renvoie une erreur EOFError dans la version de débogage de Flash Player, le message suivant s'affiche dans le champ de texte `tf` : « EOFError: Erreur #2030: Fin de fichier détectée ».

Dans une version commerciale de Flash Player ou d'Adobe AIR, le même code afficherait le texte suivant : « EOFError: Erreur #2030 ».

Remarque : *étant donné que les lecteurs de débogage diffusent l'événement « allComplete », évitez de créer des événements personnalisés portant le nom « allComplete ». Vous risquez sinon de rencontrer un comportement imprévisible lors du débogage.*

Ce type de version ne comprend pas de chaîne de message d'erreur, afin de réduire au minimum la taille et les ressources requises. Vous pouvez consulter le numéro d'erreur dans la documentation (annexes du manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash) pour l'associer à un message d'erreur. Vous pouvez également reproduire l'erreur dans les versions de débogage de Flash Player et AIR pour visualiser le message entier.

Gestion des erreurs synchrones dans une application

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La gestion des erreurs la plus courante est la logique de gestion des erreurs synchrones, qui consiste à insérer des instructions dans votre code pour intercepter les erreurs synchrones lors de l'exécution d'une application. Ce type de gestion des erreurs permet à votre application de repérer des erreurs d'exécution et de les résoudre lorsque des fonctions échouent. La logique d'interception d'une erreur synchrone fait appel aux instructions `try..catch..finally`, qui tentent littéralement une opération, interceptent toute réponse à l'erreur émanant du moteur d'exécution Flash, puis exécutent une autre opération pour gérer l'opération qui a échoué.

Utilisation des instructions try..catch..finally

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous manipulez des erreurs d'exécution synchrones, utilisez les instructions `try..catch..finally` pour intercepter les erreurs. Lorsqu'une erreur d'exécution se produit, le moteur d'exécution Flash renvoie une exception, ce qui signifie qu'il suspend l'exécution normale et crée un objet spécial de type Error. L'objet Error est ensuite renvoyé au premier bloc `catch` disponible.

L'instruction `try` regroupe les instructions pouvant créer des erreurs. Vous utilisez toujours l'instruction `catch` avec une instruction `try`. Si une erreur est détectée dans l'une des instructions du bloc `try`, les instructions `catch` associées à cette instruction `try` sont exécutées.

L'instruction `finally` regroupe les instructions exécutées, qu'une erreur se produise ou non dans le bloc `try`. S'il ne se produit pas d'erreur, les instructions du bloc `finally` sont exécutées au terme de l'exécution des instructions du bloc `try`. S'il se produit une erreur, l'instruction `catch` appropriée est exécutée en premier lieu, suivie des instructions du bloc `finally`.

Le code suivant illustre la syntaxe d'utilisation des instructions `try..catch..finally` :

```
try
{
    // some code that could throw an error
}
catch (err:Error)
{
    // code to react to the error
}
finally
{
    // Code that runs whether an error was thrown. This code can clean
    // up after the error, or take steps to keep the application running.
}
```

Chaque instruction `catch` identifie un type d'exception spécifique qu'elle gère. L'instruction `catch` peut spécifier uniquement des classes d'erreur qui sont des sous-classes de la classe Error. Chaque instruction `catch` est vérifiée dans l'ordre. Seule la première instruction `catch` qui correspond au type d'erreur renvoyé est exécutée. En d'autres termes, si vous vérifiez d'abord la classe Error de niveau supérieur, puis une sous-classe de la classe Error, seule la classe Error de niveau supérieur est prise en compte. Le code suivant illustre ce point :

```
try
{
    throw new ArgumentError("I am an ArgumentError");
}
catch (error:Error)
{
    trace("<Error> " + error.message);
}
catch (error:ArgumentError)
{
    trace("<ArgumentError> " + error.message);
}
```

Le code précédent affiche le résultat suivant :

```
<Error> I am an ArgumentError
```

Pour intercepter correctement l'erreur ArgumentError, assurez-vous que les types d'erreur les plus spécifiques sont répertoriés en premier, suivis des types d'erreur plus génériques, comme l'indique le code suivant :

```
try
{
    throw new ArgumentError("I am an ArgumentError");
}
catch (error:ArgumentError)
{
    trace("<ArgumentError> " + error.message);
}
catch (error:Error)
{
    trace("<Error> " + error.message);
}
```

Plusieurs méthodes et propriétés de l'API d'ActionScript renvoient des erreurs d'exécution si elles en rencontrent lors de leur exécution. Par exemple, la méthode `close()` de la classe Sound renvoie une erreur IOError si la méthode ne parvient pas à fermer le flux audio, comme indiqué dans le code suivant :

```
var mySound:Sound = new Sound();
try
{
    mySound.close();
}
catch (error:IOError)
{
    // Error #2029: This URLStream object does not have an open stream.
}
```

Au fur et à mesure que vous vous familiariserez avec le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash, vous identifierez les méthodes qui renvoient des exceptions, comme indiqué dans la description de chaque méthode.

Instruction throw

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les moteurs d'exécution Flash renvoient des exceptions s'ils rencontrent des erreurs lors de l'exécution de votre application. En outre, vous pouvez renvoyer des exceptions de façon explicite à l'aide de l'instruction `throw`. Si tel est le cas, Adobe vous conseille de renvoyer des occurrences de la classe Error ou de ses sous-classes. Le code suivant illustre une instruction `throw` qui renvoie une occurrence de la classe Error, `MyErr`, et appelle une fonction, `myFunction()` en réponse au renvoi de l'erreur :

```
var MyError:Error = new Error("Encountered an error with the numUsers value", 99);
var numUsers:uint = 0;
try
{
    if (numUsers == 0)
    {
        trace("numUsers equals 0");
    }
}
catch (error:uint)
{
    throw MyError; // Catch unsigned integer errors.
}
catch (error:int)
{
    throw MyError; // Catch integer errors.
}
catch (error:Number)
{
    throw MyError; // Catch number errors.
}
catch (error:*)
{
    throw MyError; // Catch any other error.
}
finally
{
    myFunction(); // Perform any necessary cleanup here.
}
```

Les instructions `catch` sont classées de façon à ce que les types de données les plus spécifiques apparaissent en premier. Si l'instruction `catch` associée au type de données Number est répertoriée en premier, ni l'instruction catch associée au type de données uint, ni l'instruction catch associée au type de données int n'est exécutée.

Remarque : *dans le langage de programmation Java, chaque fonction qui peut renvoyer une exception doit le déclarer en répertoriant les classes d'exception qu'elle peut renvoyer dans une clause `throws` associée à la déclaration de la fonction. ActionScript ne requiert pas que vous déclariez les exceptions renvoyées par une fonction.*

Affichage d'un message d'erreur simple

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'un des avantages majeurs du nouveau modèle d'événement d'erreur et d'exception consiste à permettre d'informer les utilisateurs du moment où une action échoue et de la raison de cet échec. Votre rôle consiste à écrire le code pour afficher le message et à offrir des options en réponse.

Le code suivant illustre une instruction `try..catch` simple permettant d'afficher l'erreur dans un champ de texte :

```
package
{
    import flash.display.Sprite;
    import flash.text.TextField;

    public class SimpleError extends Sprite
    {
        public var employee:XML =
            <EmpCode>
                <costCenter>1234</costCenter>
                <costCenter>1-234</costCenter>
            </EmpCode>;

        public function SimpleError()
        {
            try
            {
                if (employee.costCenter.length() != 1)
                {
                    throw new Error("Error, employee must have exactly one cost center assigned.");
                }
            }
            catch (error:Error)
            {
                var errorMessage:TextField = new TextField();
                errorMessage.autoSize = TextFieldAutoSize.LEFT;
                errorMessage.textColor = 0xFF0000;
                errorMessage.text = error.message;
                addChild(errorMessage);
            }
        }
    }
}
```

En utilisant un plus grand nombre de classes d'erreur et d'erreurs de compilateur intégrées, ActionScript 3.0 fournit de plus amples informations sur les raisons de l'échec d'une action que les versions précédentes. Ces informations permettent de créer des applications plus stables qui gèrent mieux les erreurs.

Renvoi des erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez une application, vous êtes parfois amené à renvoyer une erreur si vous ne parvenez pas à la gérer correctement. Par exemple, le code suivant illustre un bloc `try..catch` imbriqué, qui renvoie une erreur ApplicationError personnalisée si le bloc `catch` imbriqué n'est pas capable de gérer l'erreur :

```
try
{
    try
    {
        trace("<< try >>");
        throw new ApplicationError("some error which will be rethrown");
    }
    catch (error:ApplicationError)
    {
        trace("<< catch >> " + error);
        trace("<< throw >>");
        throw error;
    }
    catch (error:Error)
    {
        trace("<< Error >> " + error);
    }
}
catch (error:ApplicationError)
{
    trace("<< catch >> " + error);
}
```

Le résultat issu du fragment de code précédent serait le suivant :

```
<< try >>
<< catch >> ApplicationError: some error which will be rethrown
<< throw >>
<< catch >> ApplicationError: some error which will be rethrown
```

Le bloc `try` imbriqué renvoie une erreur ApplicationError personnalisée qui est interceptée par le bloc `catch` suivant. Ce bloc `catch` imbriqué peut tenter de gérer l'erreur et, si la tentative échoue, renvoyer l'objet ApplicationError au bloc `try..catch`.

Création de classes d'erreur personnalisées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez étendre l'une des classes Error standard pour créer vos classes d'erreur spécialisées dans ActionScript. Vous pouvez créer vos classes d'erreur pour les motifs suivants :

- Identifier des erreurs ou des groupes d'erreurs spécifiques uniques pour votre application.

  Outre les erreurs interceptées par un moteur d'exécution de Flash, vous pouvez par exemple gérer différemment les erreurs renvoyées par votre propre code. Vous pouvez créer une sous-classe de la classe Error pour suivre le nouveau type de données d'erreur dans les blocs `try..catch`.

- Fournir des fonctionnalités d'affichage d'erreurs exceptionnelles pour les erreurs générées par votre application.

  Par exemple, vous pouvez créer une méthode `toString()` qui formate vos messages d'erreur d'une certaine façon. Vous pouvez également définir une méthode `lookupErrorString()` qui prend un code d'erreur et récupère le message adéquat en fonction du langage que l'utilisateur préfère.

Une classe d'erreur spécialisée doit étendre la classe Error d'ActionScript de base. Voici un exemple de classe AppError spécialisée qui étend la classe Error :

```
public class AppError extends Error
{
    public function AppError(message:String, errorID:int)
    {
        super(message, errorID);
    }
}
```

L'exemple suivant illustre l'utilisation d'une classe AppError dans votre projet :

```
try
{
    throw new AppError("Encountered Custom AppError", 29);
}
catch (error:AppError)
{
    trace(error.errorID + ": " + error.message)
}
```

Remarque : *si vous souhaitez remplacer la méthode* `Error.toString()` *dans votre sous-classe, fournissez-lui un paramètre . . .(rest). La spécification du langage ECMAScript sur laquelle est basé ActionScript 3.0 définit ainsi la méthode* `Error.toString()` *et ActionScript 3.0 respecte cette définition à des fins de rétrocompatibilité. Par conséquent, lorsque vous remplacez la méthode* `Error.toString()`*, veillez à ce que les paramètres se correspondent exactement. Vous ne pouvez pas transmettre de paramètres à la méthode* `toString()` *lors de l'exécution, car ils sont ignorés.*

Réponse à des événements et à l'état d'erreur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'une des améliorations majeures apportées à la gestion des erreurs dans ActionScript 3.0 est la prise en charge de la gestion des événements d'erreur pour répondre à des erreurs asynchrones lors de l'exécution d'une application. (Pour obtenir une définition des erreurs asynchrones, voir « Types d'erreurs » à la page 55).

Vous pouvez créer des écouteurs d'événement et des gestionnaires d'événements pour répondre aux événements d'erreurs. De nombreuses classes distribuent des événements d'erreurs de la même façon que d'autres événements. Par exemple, une occurrence de la classe XMLSocket distribue normalement trois types d'événements :`Event.CLOSE`, `Event.CONNECT` et `DataEvent.DATA`. Néanmoins, lorsqu'un problème se produit, la classe XMLSocket peut distribuer `IOErrorEvent.IOError` ou `SecurityErrorEvent.SECURITY_ERROR`. Pour plus d'informations sur les écouteurs et les gestionnaires d'événement, voir « Gestion des événements » à la page 129.

Les événements d'erreurs peuvent être classés en deux catégories :

- Evénements d'erreurs qui étendent la classe ErrorEvent

  La classe flash.events.ErrorEvent contient les propriétés et les méthodes permettant de gérer les erreurs d'exécution liées à des opérations de réseau et de communication dans une application en cours d'exécution. Les classes AsyncErrorEvent, IOErrorEvent et SecurityErrorEvent étendent la classe ErrorEvent. Si vous utilisez la version de débogage d'un moteur d'exécution de Flash, une boîte de dialogue vous informe, lors de l'exécution, de la présence d'événements d'erreurs sans fonctions d'écouteur rencontrés par le lecteur.

- Evénements d'erreurs basés sur le statut

Les événements d'erreur basés sur le statut sont liés aux propriétés `netStatus` et `status` des classes de communication et de réseau. Si un moteur d'exécution Flash rencontre un problème lors de la lecture ou de l'écriture des données, la valeur des propriétés `netStatus.info.level` ou `status.level` (selon l'objet de classe que vous utilisez) est définie sur la valeur `"error"`. Vous répondez à cette erreur en vérifiant que la propriété `level` contient la valeur `"error"` dans votre fonction de gestionnaire d'événement.

Utilisation d'événements d'erreurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ErrorEvent et ses sous-classes contiennent des types d'erreurs destinés à gérer les erreurs distribuées par les moteurs d'exécution Flash lorsqu'ils tentent de lire ou d'écrire des données.

L'exemple suivant utilise à la fois une instruction `try..catch` et des gestionnaires d'événement d'erreur pour afficher toute erreur détectée lors de la tentative de lecture d'un fichier local. Vous pouvez ajouter un code de gestion plus élaboré pour proposer des options à l'utilisateur ou gérer l'erreur automatiquement aux endroits indiqués par le commentaire « your error-handling code here » :

```
package
{
    import flash.display.Sprite;
    import flash.errors.IOError;
    import flash.events.IOErrorEvent;
    import flash.events.TextEvent;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;

    public class LinkEventExample extends Sprite
    {
        private var myMP3:Sound;
        public function LinkEventExample()
        {
            myMP3 = new Sound();
            var list:TextField = new TextField();
            list.autoSize = TextFieldAutoSize.LEFT;
            list.multiline = true;
            list.htmlText = "<a href=\"event:track1.mp3\">Track 1</a><br>";
            list.htmlText += "<a href=\"event:track2.mp3\">Track 2</a><br>";
            addEventListener(TextEvent.LINK, linkHandler);
            addChild(list);
        }

        private function playMP3(mp3:String):void
        {
            try
            {
                myMP3.load(new URLRequest(mp3));
                myMP3.play();
            }
            catch (err:Error)
```

```
            {
                trace(err.message);
                // your error-handling code here
            }
            myMP3.addEventListener(IOErrorEvent.IO_ERROR, errorHandler);
        }

        private function linkHandler(linkEvent:TextEvent):void
        {
            playMP3(linkEvent.text);
            // your error-handling code here
        }

        private function errorHandler(errorEvent:IOErrorEvent):void
        {
            trace(errorEvent.text);
            // your error-handling code here
        }
    }
}
```

Utilisation d'événements de changement de statut

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les moteurs d'exécution Flash changent dynamiquement la valeur des propriétés `netStatus.info.level` ou `status.level` pour les classes qui prennent en charge la propriété `level` pendant qu'une application s'exécute. Les classes qui ont la propriété `netStatus.info.level` sont NetConnection, NetStream et SharedObject. Les classes qui ont la propriété `status.level` sont HTTPStatusEvent, Camera, Microphone et LocalConnection. Vous pouvez écrire une fonction de gestionnaire pour répondre au changement de valeur `level` et suivre les erreurs de communication.

L'exemple suivant utilise une fonction `netStatusHandler()` pour tester la valeur de la propriété `level`. Si la propriété `level` indique qu'une erreur a été rencontrée, le code suit le message « Video stream failed ».

```
package
{
    import flash.display.Sprite;
    import flash.events.NetStatusEvent;
    import flash.events.SecurityErrorEvent;
    import flash.media.Video;
    import flash.net.NetConnection;
    import flash.net.NetStream;

    public class VideoExample extends Sprite
    {
        private var videoUrl:String = "Video.flv";
        private var connection:NetConnection;
        private var stream:NetStream;

        public function VideoExample()
        {
            connection = new NetConnection();
            connection.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
            connection.addEventListener(SecurityErrorEvent.SECURITY_ERROR, securityErrorHandler);
            connection.connect(null);
        }
```

```
        private function netStatusHandler(event:NetStatusEvent):void
        {
            if (event.info.level == "error")
            {
                trace("Video stream failed")
            }
            else
            {
                connectStream();
            }
        }

        private function securityErrorHandler(event:SecurityErrorEvent):void
        {
            trace("securityErrorHandler: " + event);
        }

        private function connectStream():void
        {
            var stream:NetStream = new NetStream(connection);
            var video:Video = new Video();
            video.attachNetStream(stream);
            stream.play(videoUrl);
            addChild(video);
        }
    }
}
```

Comparaison des classes Error

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript fournit de nombreuses classes Error prédéfinies. Vous pouvez toutefois faire appel aux mêmes classes Error dans votre propre code. Il existe deux types principaux de classes Error dans ActionScript 3.0 : les classes Error de base d'ActionScript et les classes Error du package flash.error. Le package flash.error contient des classes supplémentaires permettant le débogage et le développement d'applications ActionScript 3.0.

Classes Error de base

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Parmi les classes Error de base figurent les classes Error, ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError et VerifyError. Chacune de ces classes se trouve dans l'espace de noms de niveau supérieur.

Nom de classe	Description	Remarques
Error	La classe Error permet de renvoyer des exceptions et correspond à la classe de base des autres classes d'exception définies dans ECMAScript : EvalError, RangeError, ReferenceError, SyntaxError, TypeError et URIError.	La classe Error sert de classe de base à toutes les erreurs d'exécution et est recommandée pour toutes les classes d'erreur personnalisées.
ArgumentError	La classe ArgumentError représente une erreur qui se produit lorsque les valeurs de paramètre fournies lors d'un appel de fonction ne correspondent pas aux paramètres définis pour celle-ci.	Voici des exemples d'erreurs d'argument : • Trop ou trop peu d'arguments sont fournis à une méthode. • Un argument devait être membre d'une énumération, et cela n'a pas été le cas.
EvalError	Une exception EvalError est renvoyée si des paramètres sont transmis au constructeur de la classe Function ou si le code utilisateur appelle la fonction `eval()`.	Dans ActionScript 3.0, la prise en charge de la fonction `eval()` a été supprimée et toute tentative d'utilisation de la fonction entraîne le renvoi d'une erreur. Les versions précédentes de Flash Player utilisaient la fonction `eval()` pour accéder à des variables, des propriétés, des objets ou des clips par nom.
RangeError	Une exception RangeError est renvoyée si une valeur numérique excède la plage acceptable.	Par exemple, une exception RangeError est renvoyée par la classe Timer si un retard est négatif ou infini. Elle peut également être renvoyée si vous tentez d'ajouter un objet d'affichage à une profondeur non valide.
ReferenceError	Une exception ReferenceError est renvoyée lorsque vous tentez d'utiliser une référence à une propriété non définie pour un objet scellé (non dynamique). Les versions du compilateur ActionScript antérieures à ActionScript 3.0 ne renvoyaient pas d'erreur lorsque vous tentiez d'accéder à une propriété `non définie`. ActionScript 3.0 renvoie toutefois l'exception ReferenceError dans ce cas de figure.	Les exceptions pour des variables non définies pointent vers des bogues éventuels afin d'améliorer la qualité du logiciel. Cependant, si vous n'avez pas l'habitude d'initialiser les variables, ce nouveau comportement d'ActionScript requiert de vous quelques changements lorsque vous écrivez du code.
SecurityError	L'exception SecurityError est renvoyée lorsqu'une violation de sécurité se produit et que l'accès est refusé.	Voici des exemples d'erreurs de sécurité : • Un accès à une propriété ou un appel de méthode non autorisé est effectué en franchissant les limites du sandbox de sécurité. • Il s'est produit une tentative d'accès à une URL non autorisée par le sandbox de sécurité. • Il s'est produit une tentative de connexion socket sur un port, mais le fichier de régulation socket approprié n'était pas présent. • Il s'est produit une tentative d'accès à la caméra ou au microphone de l'utilisateur et celui-ci a refusé l'accès au périphérique.
SyntaxError	Une exception SyntaxError est renvoyée lorsqu'il se produit une erreur d'analyse dans votre code ActionScript.	Une exception SyntaxError peut être renvoyée dans les cas suivants : • ActionScript renvoie des exceptions SyntaxError lorsque la classe RegExp analyse une expression régulière non valide. • ActionScript renvoie des exceptions SyntaxError lorsque la classe XMLDocument analyse un code XML non valide.

Nom de classe	Description	Remarques
TypeError	L'exception TypeError est renvoyée lorsque le type réel d'un opérande ne correspond pas au type prévu.	Une exception TypeError peut être renvoyée dans les cas suivants : • Un paramètre réel de fonction ou de méthode ne peut pas être forcé à correspondre au type de paramètre formel. • Une valeur est affectée à une variable et ne peut pas être forcée à correspondre au type de la variable. • Le côté droit de l'opérateur `is` ou `instanceof` n'est pas un type valide. • L'utilisation du mot clé `super` n'est pas valide. • Une recherche de propriété donne lieu à plusieurs liaisons, soit un résultat ambigu. • Une méthode est appelée pour un objet incompatible. Par exemple, une exception TypeError est renvoyée si une méthode de la classe RegExp est « greffée » sur un objet générique, puis appelée.
URIError	L'exception URIError est renvoyée lorsque l'une des fonctions de gestion URI globales est utilisée d'une manière qui n'est pas compatible avec sa définition.	Une exception URIError peut être renvoyée dans les cas suivants : Un URI non valide est défini par une fonction API de Flash Player qui s'attend à un URI valide, comme `Socket.connect()`.
VerifyError	Une erreur VerifyError est renvoyée lorsqu'un fichier SWF incorrect ou altéré est détecté.	Lorsqu'un fichier SWF charge un autre fichier SWF, le fichier SWF parent peut intercepter une exception VerifyError générée par le fichier SWF chargé.

Classes Error du package flash.error

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le package flash.error contient des classes Error considérées comme parties intégrantes de l'API des moteurs d'exécution de Flash. A l'encontre des classes Error décrites, le package flash.error communique les événements d'erreurs propres aux moteurs d'exécution de Flash (tel Flash Player ou Adobe AIR).

Nom de classe	Description	Remarques
EOFError	Une exception EOFError est émise lors d'une tentative de lecture au-delà de la fin des données disponibles.	Par exemple, une exception EOFError est émise chaque fois qu'une méthode de lecture de l'interface IDataInput est appelée et que les données sont insuffisantes pour répondre à la requête de lecture.
IllegalOperationError	Une exception IllegalOperationError est renvoyée lorsqu'une méthode n'est pas implémentée ou si l'implémentation ne couvre pas l'utilisation actuelle.	Voici quelques exemples d'exceptions d'erreurs liées à des opérations non valides : • Une classe de base, telle que DisplayObjectContainer, propose plus de fonctionnalités qu'une scène ne peut prendre en charge. Par exemple, si vous tentez d'obtenir ou de définir un masque sur la scène (à l'aide de `stage.mask`), le moteur d'exécution de Flash renvoie une exception IllegalOperationError accompagnée du message « La classe Stage n'implémente ni cette propriété, ni cette méthode ». • Une sous-classe hérite d'une méthode dont elle n'a pas besoin et qu'elle ne souhaite pas prendre en charge. • Certaines méthodes d'accessibilité sont appelées lorsque Flash Player est compilé sans les fonctions d'accessibilité. • Les fonctions réservées à la création sont appelées à partir d'une version d'exécution de Flash Player. • Vous tentez de définir le nom d'un objet placé sur le scénario.
IOError	Une exception IOError est renvoyée lorsqu'un type d'exception E/S se produit.	Vous obtenez cette erreur, par exemple, lorsque vous tentez une opération de lecture-écriture sur un socket qui n'est pas connecté ou qui est déconnecté.
MemoryError	Une exception MemoryError est renvoyée lors de l'échec d'une requête d'allocation de mémoire.	Par défaut, ActionScript Virtual Machine 2 n'impose pas de limite à la quantité de mémoire allouée par un programme ActionScript. Sur un système de bureau, les échecs d'allocation de mémoire sont rares. Une erreur est renvoyée lorsque le système ne parvient pas à allouer la mémoire requise pour une opération. Par conséquent, sur un système de bureau, cette exception est peu fréquente, à moins qu'une requête d'allocation ne soit extrêmement importante (par exemple, une requête de 3 milliards d'octets est impossible, car un programme Microsoft® Windows® de 32 bits peut accéder à 2 Go d'espace d'adressage uniquement).
ScriptTimeoutError	Une exception ScriptTimeoutError est renvoyée lorsqu'un intervalle de délai d'expiration du script de 15 secondes est atteint. En interceptant une exception ScriptTimeoutError, vous pouvez gérer le délai d'expiration du script plus en douceur. Si aucun gestionnaire d'exception n'est défini, le gestionnaire de l'exception non interceptée affiche une boîte de dialogue contenant un message d'erreur.	Pour éviter qu'un développeur n'intercepte l'exception et reste dans une boucle sans fin, seule la première exception ScriptTimeoutError renvoyée au cours d'un script donné peut être interceptée. Le code ne peut pas intercepter une exception ScriptTimeoutError ultérieure. Elle passe donc immédiatement au gestionnaire de l'exception non interceptée.
StackOverflowError	L'exception StackOverflowError est renvoyée lorsque la pile disponible pour le script a été épuisée.	Une exception StackOverflowError peut indiquer qu'un problème de récursivité à l'infini s'est produit.

Exemple de gestion des erreurs : application CustomErrors

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application CustomErrors décrit les techniques d'utilisation des erreurs personnalisées lors de la création d'une application. Ces techniques sont les suivantes :

- Validation d'un paquet XML
- Ecriture d'une erreur personnalisée
- Renvoi d'erreurs personnalisées
- Notification des utilisateurs lors du renvoi d'une erreur

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application CustomErrors se trouvent dans le dossier Samples/CustomError. L'application se compose des fichiers suivants :

Fichier	Description
CustomErrors.mxml ou CustomErrors.fla	Fichier d'application principal en FLA pour Flash ou en MXML pour Flex
com/example/programmingas3/errors/ApplicationError.as	Une classe servant de classe Error de base pour les classes FatalError et WarningError.
com/example/programmingas3/errors/FatalError.as	Une classe qui définit une erreur FatalError renvoyée par l'application. Cette classe étend la classe ApplicationError personnalisée.
com/example/programmingas3/errors/Validator.as	Une classe qui définit une seule méthode qui valide un paquet XML employee fourni par l'utilisateur.
com/example/programmingas3/errors/WarningError.as	Une classe qui définit une erreur WarningError renvoyée par l'application. Cette classe étend la classe ApplicationError personnalisée.

Présentation de l'application CustomErrors

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Au chargement de l'application, la méthode `initApp()` est appelée dans les applications Flex ou le code du scénario (autre qu'une fonction) est exécuté dans les applications Flash Professional. Ce code définit un exemple de paquet XML que la classe Validator vérifiera. Le code suivant est exécuté :

```
employeeXML =
    <employee id="12345">
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <costCenter>12345</costCenter>
        <costCenter>67890</costCenter>
    </employee>;
}
```

Le paquet XML est ensuite affiché dans une occurrence du composant TextArea sur la scène. Cette étape permet de modifier le paquet XML avant de tenter de le revalider.

Lorsque l'utilisateur clique sur le bouton de validation, la méthode `validateData()` est appelée. Cette méthode valide le paquet XML employee à l'aide de la méthode `validateEmployeeXML()` de la classe Validator. Le code suivant présente la méthode `validateData()` :

```
function validateData():void
{
    try
    {
        var tempXML:XML = XML(xmlText.text);
        Validator.validateEmployeeXML(tempXML);
        status.text = "The XML was successfully validated.";
    }
    catch (error:FatalError)
    {
        showFatalError(error);
    }
    catch (error:WarningError)
    {
        showWarningError(error);
    }
    catch (error:Error)
    {
        showGenericError(error);
    }
}
```

Un objet XML temporaire est d'abord créé à l'aide du contenu de l'occurrence du composant TextArea `xmlText`. La méthode `validateEmployeeXML()` de la classe Validator personnalisée (com.example.programmingas3/errors/Validator.as) est ensuite appelée et transmet l'objet XML temporaire comme paramètre. Si le paquet XML est valide, l'occurrence du composant Label `status` affiche un message de réussite et l'application se ferme. Si la méthode `validateEmployeeXML()` renvoie une erreur personnalisée (c'est-à-dire qu'une erreur FatalError, WarningError ou une erreur générique se produit), l'instruction `catch` appropriée s'exécute et appelle les méthodes `showFatalError()`, `showWarningError()` ou `showGenericError()`. Chacune de ces méthodes affiche un message approprié dans une zone de texte appelée `statusTex`t pour avertir l'utilisateur de l'erreur spécifique qui s'est produite. Chaque méthode met également à jour l'occurrence du composant Label `status` avec un message spécifique.

Si une erreur grave se produit pendant une tentative de validation du paquet XML employee, le message d'erreur s'affiche dans la zone de texte `statusText` et l'occurrence du composant TextArea `xmlText` et l'occurrence du composant Button `validateBtn` sont désactivées, comme l'indique le code suivant :

```
function showFatalError(error:FatalError):void
{
    var message:String = error.message + "\n\n";
    var title:String = error.getTitle();
    statusText.text = message + " " + title + "\n\nThis application has ended.";
    this.xmlText.enabled = false;
    this.validateBtn.enabled = false;
    hideButtons();
}
```

S'il se produit une erreur d'avertissement au lieu d'une erreur grave, le message d'erreur est affiché dans l'occurrence de TextArea `statusText`, mais les occurrences de composant TextField `xmlText` et Button ne sont pas désactivées. La méthode `showWarningError()` affiche le message d'erreur personnalisé dans la zone de texte `statusText`. Le message invite également l'utilisateur à indiquer s'il souhaite poursuivre la validation de l'objet XML ou annuler le script. Le fragment de code suivant présente la méthode `showWarningError()` :

```
function showWarningError(error:WarningError):void
{
    var message:String = error.message + "\n\n" + "Do you want to exit this application?";
    showButtons();
    var title:String = error.getTitle();
    statusText.text = message;
}
```

Lorsque l'utilisateur clique sur le bouton Oui ou Non, la méthode `closeHandler()` est appelée. Le fragment de code suivant présente la méthode `closeHandler()` :

```
function closeHandler(event:CloseEvent):void
{
    switch (event.detail)
    {
        case yesButton:
            showFatalError(new FatalError(9999));
            break;
        case noButton:
            statusText.text = "";
            hideButtons();
            break;
    }
}
```

Si l'utilisateur souhaite annuler le script en cliquant sur Oui, une exception FatalError est renvoyée, provoquant l'arrêt de l'application.

Création d'une validation personnalisée

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Validator personnalisée contient une seule méthode, `validateEmployeeXML()`. La méthode `validateEmployeeXML()` gère un seul argument, `employee`, qui correspond au paquet XML à valider. La méthode `validateEmployeeXML()` est la suivante :

```
public static function validateEmployeeXML(employee:XML):void
{
    // checks for the integrity of items in the XML
    if (employee.costCenter.length() < 1)
    {
        throw new FatalError(9000);
    }
    if (employee.costCenter.length() > 1)
    {
        throw new WarningError(9001);
    }
    if (employee.ssn.length() != 1)
    {
        throw new FatalError(9002);
    }
}
```

Un employé doit appartenir à un (et un seul) centre de coût pour être validé. S'il n'appartient à aucun centre de coût, la méthode renvoie une erreur FatalError, qui se propage jusqu'à la méthode `validateData()` dans le fichier d'application principale. Si l'employé appartient à plusieurs centres de coût, une erreur WarningError est renvoyée. La dernière vérification de la validation XML contrôle que l'utilisateur possède un seul numéro de sécurité sociale défini (le nœud `ssn` dans le paquet XML). S'il n'y a pas exactement un nœud `ssn`, une erreur FatalError est renvoyée.

Vous pouvez ajouter des vérifications supplémentaires à la méthode `validateEmployeeXML()` afin de vérifier, par exemple, que le nœud `ssn` contient un numéro valide, ou que l'employé possède au moins un numéro de téléphone et une adresse électronique, et que ces deux valeurs sont valides. Vous pouvez également modifier le XML de façon à ce que chaque employé ait un ID unique et spécifie l'ID de son responsable.

Définition de la classe ApplicationError

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ApplicationError sert de classe de base aux classes FatalError et WarningError. Elle étend la classe Error et définit ses propres propriétés et méthodes personnalisées, y compris un ID d'erreur, la gravité et un objet XML contenant les codes d'erreur personnalisés et les messages. Cette classe définit également deux constantes statiques utilisées pour définir la gravité de chaque type d'erreur.

La méthode du constructeur de la classe ApplicationError est la suivante :

```
public function ApplicationError()
{
    messages =
        <errors>
            <error code="9000">
                <![CDATA[Employee must be assigned to a cost center.]]>
            </error>
            <error code="9001">
                <![CDATA[Employee must be assigned to only one cost center.]]>
            </error>
            <error code="9002">
                <![CDATA[Employee must have one and only one SSN.]]>
            </error>
            <error code="9999">
                <![CDATA[The application has been stopped.]]>
            </error>
        </errors>;
}
```

Chaque nœud d'erreur dans l'objet XML contient un code numérique unique et un message d'erreur. Vous pouvez consulter facilement les messages d'erreur par code d'erreur à l'aide d'E4X, comme indiqué dans la méthode `getMessageText()` suivante :

```
public function getMessageText(id:int):String
{
    var message:XMLList = messages.error.(@code == id);
    return message[0].text();
}
```

La méthode `getMessageText()` prend un seul argument entier, `id`, et renvoie une chaîne. L'argument `id` correspond au code de l'erreur à rechercher. Par exemple, lorsque vous transmettez un `id` de 9001, l'erreur indiquant que les employés doivent être affectés à un seul centre de coût est renvoyée. Si plusieurs erreurs ont le même code, ActionScript renvoie le message d'erreur uniquement pour le premier résultat trouvé (`message[0]` dans l'objet XMLList renvoyé).

La méthode suivante de cette classe, `getTitle()`, ne prend aucun paramètre et renvoie une valeur de chaîne qui contient l'ID de cette erreur spécifique. Cette valeur permet d'identifier aisément l'erreur exacte qui s'est produite lors de la validation du paquet XML. Le fragment de code suivant présente la méthode `getTitle()` :

```
public function getTitle():String
{
    return "Error #" + id;
}
```

La dernière méthode finale de la classe ApplicationError est `toString()`. Cette méthode remplace la fonction définie dans la classe Error pour que vous puissiez personnaliser la présentation du message d'erreur. La méthode renvoie une chaîne qui identifie le numéro d'erreur spécifique et le message qui s'est affiché.

```
public override function toString():String
{
    return "[APPLICATION ERROR #" + id + "] " + message;
}
```

Définition de la classe FatalError

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe FatalError étend la classe ApplicationError personnalisée et définit trois méthodes : le constructeur FatalError, `getTitle()` et `toString()`. La première méthode, le constructeur FatalError, prend un seul argument entier, `errorID`, et définit la gravité de l'erreur à l'aide des valeurs de constante statiques définies dans la classe ApplicationError. Elle obtient le message de l'erreur spécifique en appelant la méthode `getMessageText()` dans la classe ApplicationError. Le constructeur FatalError se présente comme suit :

```
public function FatalError(errorID:int)
{
    id = errorID;
    severity = ApplicationError.FATAL;
    message = getMessageText(errorID);
}
```

La méthode suivante de la classe FatalError, `getTitle()`, remplace la méthode `getTitle()` définie précédemment dans la classe ApplicationError et ajoute le texte "-- FATAL" dans le titre pour informer l'utilisateur qu'une erreur grave s'est produite. La méthode `getTitle()` se présente comme suit :

```
public override function getTitle():String
{
    return "Error #" + id + " -- FATAL";
}
```

La méthode finale dans cette classe, `toString()`, remplace la méthode `toString()` définie dans la classe ApplicationError. La méthode `toString()` est la suivante :

```
public override function toString():String
{
    return "[FATAL ERROR #" + id + "] " + message;
}
```

Définition de la classe WarningError

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe WarningError étend la classe ApplicationError et est presque identique à la classe FatalError, à l'exception de quelques changements de chaîne mineurs. Elle définit la gravité de l'erreur sur ApplicationError.WARNING au lieu de ApplicationError.FATAL, comme indiqué dans le code suivant :

```
public function WarningError(errorID:int)
{
    id = errorID;
    severity = ApplicationError.WARNING;
    message = super.getMessageText(errorID);
}
```

Chapitre 5 : Utilisation d'expressions régulières

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une expression régulière décrit un modèle servant à rechercher et à manipuler du texte de correspondance dans des chaînes. Les expressions régulières ressemblent à des chaînes, mais elles comprennent des codes spéciaux pour décrire des modèles et des répétitions. Par exemple, l'expression régulière suivante correspond à une chaîne qui commence par le caractère A suivi d'un ou de plusieurs chiffres séquentiels :

```
/A\d+/
```

Les rubriques suivantes sont consacrées à la syntaxe de base de construction d'expressions régulières. Néanmoins, les expressions régulières peuvent être très complexes et comporter de nombreuses nuances. Vous pouvez vous documenter sur les expressions régulières sur le Web et dans les librairies. Différents environnements de programmation implémentent des expressions régulières de différentes façons. ActionScript 3.0 implémente des expressions régulières comme défini dans la version 3 de la spécification du langage ECMAScript (ECMA-262).

Voir aussi

RegExp

Principes de base des expressions régulières

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une expression régulière décrit un modèle de caractères. Les expressions régulières servent généralement à vérifier qu'une valeur de texte est conforme à un modèle particulier (par exemple, vérifier qu'un numéro de téléphone saisi par l'utilisateur comporte le nombre de chiffres correct) ou à remplacer des portions d'une valeur de texte qui correspondent à un modèle donné.

Les expressions régulières peuvent être simples. Par exemple, supposons que vous souhaitiez confirmer qu'une chaîne particulière correspond à ABC ou que vous souhaitiez remplacer chaque occurrence d'ABC dans une chaîne par un autre texte. Dans ce cas, vous pouvez utiliser l'expression régulière suivante qui définit le modèle comportant les lettres A, B et C, dans l'ordre :

```
/ABC/
```

Le littéral de l'expression régulière est délimité avec la barre oblique (`/`).

Les modèles d'expression régulière peuvent également être complexes et parfois sembler obscures, comme l'expression suivante pour établir une correspondance avec une adresse électronique valide :

```
/([0-9a-zA-Z]+[-._+&])*[0-9a-zA-Z]+@([-0-9a-zA-Z]+[.])+[a-zA-Z]{2,6}/
```

Vous utiliserez le plus souvent des expressions régulières pour rechercher des modèles dans des chaînes et pour remplacer des caractères. Dans ces cas, vous créerez un objet d'expression régulière et l'utiliserez comme paramètre pour une ou plusieurs méthodes de classe String. Les méthodes suivantes de la classe String prennent des expressions régulières comme paramètres : `match()`, `replace()`, `search()` et `split()`. Pour plus d'informations sur ces méthodes, voir « Recherche de modèles dans des chaînes et remplacement de sous-chaînes » à la page 17.

La classe RegExp comprend les méthodes suivantes : `test()` et `exec()`. Pour plus d'informations, voir « Méthodes d'utilisation d'expressions régulières avec des chaînes » à la page 93.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la fonctionnalité étudiée.

Caractère d'échappement Caractère indiquant que le caractère qui suit doit être considéré comme un caractère de remplacement plutôt que comme un caractère littéral. Dans une syntaxe d'expression régulière, la barre oblique inverse (\) est le caractère d'échappement. Par conséquent, une barre oblique inverse suivie d'un autre caractère est un code spécial plutôt que le caractère à proprement parler.

Indicateur Caractère qui spécifie une option associée au mode d'utilisation du modèle d'expression régulière (respect de la casse, par exemple).

Caractère de remplacement Caractère qui a une signification spéciale dans un modèle d'expression régulière et qui ne représente pas littéralement ce caractère dans le modèle.

Quantificateur Caractères indiquant le nombre de répétitions d'une partie du modèle. Par exemple, un quantificateur peut être utilisé pour indiquer qu'un code postal américain doit contenir cinq ou neuf chiffres.

Expression régulière Instruction de programme définissant un modèle de caractères qui permet de confirmer si d'autres chaînes correspondent à ce modèle ou de remplacer des sections d'une chaîne.

Syntaxe d'expression régulière

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cette section décrit tous les éléments de la syntaxe d'expression régulière d'ActionScript. Comme vous pourrez le constater, les expressions régulières peuvent être très complexes et comporter de nombreuses nuances. Vous pouvez vous documenter sur les expressions régulières sur le Web et dans les librairies. Différents environnements de programmation implémentent des expressions régulières de différentes façons. ActionScript 3.0 implémente des expressions régulières comme défini dans la version 3 de la spécification du langage ECMAScript (ECMA-262).

Généralement, vous utilisez des expressions régulières qui correspondent à des modèles plus compliqués qu'une simple chaîne de caractères. Par exemple, l'expression régulière suivante définit le modèle comportant les lettres A, B et C, dans l'ordre, suivies par un chiffre :

```
/ABC\d/
```

Le code `\d` représente un chiffre. La barre oblique inverse (`\`) est appelée caractère d'échappement. Lorsqu'elle est combinée au caractère qui la suit (dans ce cas, la lettre d), elle a une signification spéciale dans l'expression régulière.

L'expression régulière suivante définit le modèle des lettres ABC suivies par des chiffres (remarquez l'astérisque) :

```
/ABC\d*/
```

L'astérisque (*) est un *caractère de remplacement*. Un caractère de remplacement est un caractère ayant une signification spéciale dans les expressions régulières. L'astérisque est un type de caractère de remplacement spécifique appelé *quantificateur,* utilisé pour quantifier le nombre de répétitions d'un caractère ou groupe de caractères. Pour plus de détails, voir « Quantificateurs » à la page 85.

Outre son modèle, une expression régulière peut contenir des indicateurs qui spécifient comment l'expression régulière doit être mise en correspondance. Par exemple, l'expression régulière suivante utilise l'indicateur `i` qui indique qu'elle ignore le respect de la casse dans les chaînes de correspondance :

```
/ABC\d*/i
```

Pour plus d'informations, voir « Indicateurs et propriétés » à la page 89.

Vous pouvez utiliser des expressions régulières à l'aide des méthodes suivantes de la classe String : `match()`, `replace()` et `search()`. Pour plus d'informations sur ces méthodes, voir « Recherche de modèles dans des chaînes et remplacement de sous-chaînes » à la page 17.

Création d'une occurrence d'expression régulière

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il existe deux façons de créer une occurrence d'expression régulière. L'une des méthodes consiste à utiliser des barres obliques (/) pour délimiter l'expression régulière, et l'autre consiste à utiliser le constructeur `new`. Par exemple, les expressions régulières suivantes sont équivalentes :

```
var pattern1:RegExp = /bob/i;
var pattern2:RegExp = new RegExp("bob", "i");
```

Des barres obliques délimitent un littéral d'expression régulière de la même façon que des guillemets délimitent un littéral de chaîne. La partie de l'expression régulière contenue entre les barres obliques définit le *modèle.* L'expression régulière peut également inclure des *indicateurs* après la barre de délimitation finale. Ces indicateurs sont considérés comme faisant partie de l'expression régulière, mais ils sont séparés de son modèle.

Lorsque vous utilisez le constructeur `new`, vous utilisez deux chaînes pour définir l'expression régulière. La première chaîne définit le modèle, et la seconde les indicateurs, comme dans l'exemple suivant :

```
var pattern2:RegExp = new RegExp("bob", "i");
```

Lorsque vous incluez une barre oblique *dans* une expression régulière qui est définie à l'aide de délimiteurs de barre oblique, vous devez faire précéder la barre oblique du caractère d'échappement (\). Par exemple, l'expression régulière suivante correspond au modèle `1/2` :

```
var pattern:RegExp = /1\/2/;
```

Pour inclure des guillemets *dans* une expression régulière définie avec le constructeur `new`, vous devez ajouter un caractère d'échappement (\) avant les guillemets (comme lorsque vous définissez un littéral String). Par exemple, les expressions régulières suivantes correspondent au modèle `eat at "joe's"` :

```
var pattern1:RegExp = new RegExp("eat at \"joe's\"", "");
var pattern2:RegExp = new RegExp('eat at "joe\'s"', "");
```

N'utilisez pas le caractère d'échappement avec des guillemets dans des expressions régulières définies à l'aide des délimiteurs de barre oblique. De même, n'utilisez pas le caractère d'échappement avec des barres obliques dans des expressions régulières définies avec le constructeur `new`. Les expressions régulières suivantes sont équivalentes. Elles définissent le modèle `1/2 "joe's"` :

```
var pattern1:RegExp = /1\/2 "joe's"/;
var pattern2:RegExp = new RegExp("1/2 \"joe's\"", "");
var pattern3:RegExp = new RegExp('1/2 "joe\'s"', '');
```

De même, dans une expression régulière définie à l'aide du constructeur `new`, tapez deux fois le caractère barre oblique inverse pour utiliser une métaséquence débutant par le caractère barre oblique inverse(`\`), telle que `\d` (qui correspond à n'importe quel chiffre).

```
var pattern:RegExp = new RegExp("\\d+", ""); // matches one or more digits
```

Vous devez taper deux fois le caractère barre oblique inverse, car le premier paramètre de la méthode constructeur `RegExp()` est une chaîne. Dans un littéral de chaîne, ce caractère doit être entré deux fois pour être interprété comme une barre oblique inverse unique.

La section qui suit décrit la syntaxe servant à définir des modèles d'expression régulière.

Pour plus d'informations, voir « Indicateurs et propriétés » à la page 89.

Caractères, caractères de remplacement et métaséquences

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'expression régulière la plus simple est celle qui correspond à une séquence de caractères, comme dans l'exemple suivant :

```
var pattern:RegExp = /hello/;
```

Néanmoins, les caractères suivants, appelés caractères de remplacement, ont des significations spéciales dans des expressions régulières :

```
^ $ \ . * + ? ( ) [ ] { } |
```

Par exemple, l'expression régulière suivante correspond à la lettre A suivie par zéro ou plusieurs occurrences de la lettre B (le caractère de remplacement astérisque indique cette répétition), suivie par la lettre C :

```
/AB*C/
```

Pour inclure un caractère de remplacement sans sa signification spéciale dans un modèle d'expression régulière, vous devez utiliser le caractère d'échappement (`\`). Par exemple, l'expression régulière suivante correspond à la lettre A suivie par la lettre B, suivie par un astérisque, suivie par la lettre C :

```
var pattern:RegExp = /AB\*C/;
```

Une *métaséquence,* comme un caractère de remplacement, a une signification spéciale dans une expression régulière. Une métaséquence est constituée de plusieurs caractères. Les sections suivantes fournissent des détails sur l'utilisation des caractères de remplacement et des métaséquences.

A propos des caractères de remplacement

Le tableau suivant répertorie les caractères de remplacement que vous pouvez utiliser dans des expressions régulières :

Caractère de remplacement	Description
^ (caret)	Etablit une correspondance au début d'une chaîne. Lorsque l'indicateur `m` (`multiline`) est défini, le caret correspond au début d'une ligne également (voir « Indicateurs et propriétés » à la page 89). Lorsqu'il est utilisé au début d'une classe de caractère, le caret indique la négation et non le début d'une chaîne. Pour plus d'informations, voir « Classes de caractère » à la page 83.
$(signe dollar)	Etablit une correspondance à la fin d'une chaîne. Lorsque l'indicateur `m` (`multiline`) est défini, `$` correspond à la position avant une nouvelle ligne (`\n`) également. Pour plus d'informations, voir « Indicateurs et propriétés » à la page 89.
\ (caractère d'échappement)	Echappe la signification spéciale du caractère de remplacement des caractères spéciaux. Vous pouvez également utiliser le caractère d'échappement si vous souhaitez utiliser une barre oblique dans un littéral d'expression régulière, comme dans `/1\/2/` (pour correspondre au caractère 1, suivi par la barre oblique, suivi par le caractère 2).
. (point)	Correspond à un seul caractère. Un point correspond à un caractère de nouvelle ligne (`\n`) uniquement si l'indicateur `s` (`dotall`) est défini. Pour plus d'informations, voir « Indicateurs et propriétés » à la page 89.
* (étoile)	Correspond à l'élément précédent répété zéro ou plusieurs fois. Pour plus de détails, voir « Quantificateurs » à la page 85.
+ (plus)	Correspond à l'élément précédent répété une ou plusieurs fois. Pour plus de détails, voir « Quantificateurs » à la page 85.
? (point d'interrogation)	Correspond à l'élément précédent répété zéro ou une fois. Pour plus de détails, voir « Quantificateurs » à la page 85.
(et)	Définit des groupes dans l'expression régulière. Utilisez des groupes pour : • confiner le domaine du permutateur | : `/(a|b|c)d/` • définir le domaine d'un quantificateur :`/(walla.){1,2}/` • dans des backreferences. Par exemple, le `\1` dans l'expression régulière suivante correspond à toute correspondance au premier groupe entre parenthèses du modèle : • `/(\w*) est répété : \1/` Pour plus d'informations, voir « Groupes » à la page 87.
[et]	Spécifie une classe de caractère qui définit des correspondances possibles pour un seul caractère : `/[aeiou]/` correspond à l'un des caractères spécifiés. Dans les classes de caractère, utilisez le trait d'union (-) pour désigner une plage de caractères : `/[A-Z0-9]/` correspond aux lettres majuscules A à Z ou aux chiffres 0 à 9. Dans les classes de caractère, insérez un caractère d'échappement pour échapper les caractères] et les caractères -: `/[+\-]\d+/` correspond à + ou à - avant un ou plusieurs chiffres. Dans les classes de caractère, d'autres caractères (qui sont normalement des caractères de remplacement) sont considérés comme des caractères normaux (et non comme des caractères de remplacement), sans qu'une barre oblique inverse soit nécessaire : `/[$]/`£ correspond à `$ou à` £. Pour plus d'informations, voir « Classes de caractère » à la page 83.
| *(opérateur de transfert de données)*	Utilisé pour la permutation, pour correspondre à la partie de gauche ou à celle de droite : `/abc|xyz/` correspond à `abc` ou à `xyz`.

A propos des métaséquences

Les métaséquences sont des séquences de caractères ayant une signification spéciale dans un modèle d'expression régulière. Le tableau suivant décrit ces métaséquences :

Métaséquence	Description
{*n*} {*n*,} et {*n*,*n*}	Indique un quantificateur numérique ou une plage de quantificateurs pour l'élément précédent : `/A{27}/` correspond au caractère `A` répété `27` fois. `/A{3}/` correspond au caractère `A` répété `3` fois ou plus. `/A{3,5}/` correspond au caractère `A` répété `3` à `5` fois. Pour plus de détails, voir « Quantificateurs » à la page 85.
`\b`	Etablit une correspondance à la position entre un caractère mot et un caractère non-mot. Si le premier ou le dernier caractère dans la chaîne est un caractère mot, correspond également au début ou à la fin de la chaîne.
`\B`	Etablit une correspondance à la position entre deux caractères mot. Correspond également à la position entre deux caractères non-mot.
`\d`	Correspond à une décimale.
`\D`	Correspond à tout caractère autre qu'un chiffre.
`\f`	Correspond à un caractère de changement de page.
`\n`	Correspond au caractère de nouvelle ligne.
`\r`	Correspond au caractère de retour de chariot.
`\s`	Correspond à tout caractère d'espace blanc (un espace, une tabulation, une nouvelle ligne ou un caractère de retour de chariot).
`\S`	Correspond à tout caractère autre qu'un caractère d'espace blanc.
`\t`	Correspond au caractère de tabulation.
`\u`*nnnn*	Correspond au caractère Unicode avec le code de caractère spécifié par le nombre hexadécimal *nnnn*. Par exemple, `\u263a` est le caractère smiley.
`\v`	Correspond à un caractère d'avancement vertical.
`\w`	Correspond à un caractère mot (`A`–`Z`, `a`–`z`, `0`–`9` ou `_`). `\w` ne correspond pas aux caractères qui ne sont pas anglais tels que `é`, `ñ`, ou `ç`.
`\W`	Correspond à tout caractère autre qu'un caractère mot.
`\\x`*nn*	Correspond au caractère avec la valeur ASCII spécifiée, comme défini par le nombre hexadécimal *nn*.

Classes de caractère

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous utilisez des classes de caractère pour spécifier une liste de caractères devant correspondre à une position dans l'expression régulière. Vous définissez des classes de caractère avec des crochets (`[` et `]`). Par exemple, l'expression régulière suivante définit une classe de caractère qui remplace `bag`, `beg`, `big`, `bog` ou `bug` :

```
/b[aeiou]g/
```

Séquences d'échappement dans des classes de caractère

La plupart des caractères de remplacement et des métaséquences ayant normalement des significations spéciales dans une expression régulière *n'ont pas* ces mêmes significations dans une classe de caractère. Par exemple, dans une expression régulière, l'astérisque est utilisé pour la répétition, mais ce n'est pas le cas lorsqu'il apparaît dans une classe de caractère. La classe de caractère suivante correspond à l'astérisque de façon littérale, avec tout autre caractère répertorié :

```
/[abc*123]/
```

Cependant, les trois caractères répertoriés dans le tableau suivant fonctionnent comme des caractères de remplacement, avec une signification spéciale, dans des classes de caractère :

Caractère de remplacement	Signification dans des classes de caractère
`]`	Définit la fin de la classe de caractère.
`-`	Définit une plage de caractères (voir la section suivante, " Plages de caractères dans des classes de caractère ").
`\`	Définit des métaséquences et annule la signification spéciale des caractères de remplacement.

Pour que ces caractères soient reconnus comme caractères littéraux (dans la signification du caractère de remplacement spéciale), vous devez faire précéder le caractère du caractère d'échappement. Par exemple, l'expression régulière suivante inclut une classe de caractère qui correspond à l'un des quatre symboles (`$`, `\`, `]` ou `-`) :

```
/[$\\\]\-]/
```

Outre les caractères de remplacement qui conservent leurs significations spéciales, les métaséquences suivantes fonctionnent comme des métaséquences dans des classes de caractère :

Métaséquence	Signification dans des classes de caractère
`\n`	Correspond à un caractère de nouvelle ligne.
`\r`	Correspond à un caractère de retour de chariot.
`\t`	Correspond à un caractère de tabulation.
`\u`***nnnn***	Correspond au caractère avec la valeur de point de code Unicode spécifiée (comme défini par le nombre hexadécimal *nnnn*).
`\\x`***nn***	Correspond au caractère avec la valeur ASCII spécifiée (comme défini par le nombre hexadécimal *nn*).

D'autres caractères de remplacement et métaséquences d'expression régulière sont considérés comme des caractères normaux dans une classe de caractère.

Plages de caractères dans des classes de caractère

Utilisez le trait d'union pour spécifier une plage de caractères telle que `A-Z`, `a-z`, ou `0-9`. Ces caractères doivent constituer une plage valide dans le jeu de caractères. Par exemple, la classe de caractère suivante correspond à un caractère compris dans la plage `a-z` ou un chiffre :

```
/[a-z0-9]/
```

Vous pouvez également utiliser le code de caractère ASCII `\\x`*nn* pour spécifier une plage par valeur ASCII. Par exemple, la classe de caractère suivante correspond à un caractère d'un jeu de caractères ASCII étendus (é et ê, par exemple) :

```
\\x
```

Classes de caractère niées

Lorsque vous utilisez un caret (^) au début d'une classe de caractère, il la nie (tout caractère non répertorié est considéré comme une correspondance). La classe de caractère suivante correspond à tout caractère *sauf* une lettre minuscule (az–) ou un chiffre :

```
/[^a-z0-9]/
```

Vous devez taper le caret (^) au *début* d'une classe de caractère pour indiquer la négation. Autrement, vous ajoutez simplement le caret aux caractères dans la classe de caractère. Par exemple, la classe de caractère suivante correspond à un symbole (le caret, notamment) :

```
/[!.,#+*%$&^]/
```

Quantificateurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous utilisez des quantificateurs pour spécifier des répétitions de caractères ou de séquences dans des modèles, comme suit :

Caractère de remplacement de quantificateur	Description
* (étoile)	Correspond à l'élément précédent répété zéro ou plusieurs fois.
+ (plus)	Correspond à l'élément précédent répété une ou plusieurs fois.
? (point d'interrogation)	Correspond à l'élément précédent répété zéro ou une fois.
{***n***} {***n***,} et {***n***,***n***}	Indique un quantificateur numérique ou une plage de quantificateurs pour l'élément précédent : /A{27}/ correspond au caractère A répété 27 fois. /A{3}/ correspond au caractère A répété 3 fois ou plus. /A{3,5}/ correspond au caractère A répété 3 à 5 fois.

Vous pouvez appliquer un quantificateur à un seul caractère, à une classe de caractère ou à un groupe :

- /a+/ correspond au caractère a répété une ou plusieurs fois.
- /\d+/ correspond à un ou plusieurs chiffres.
- /[abc]+/ correspond à une répétition d'un ou de plusieurs caractères, a, b, ou c.
- /(very,)*/ correspond au mot very suivi par une virgule et un espace répété zéro ou plusieurs fois.

Vous pouvez utiliser des quantificateurs dans des groupes de parenthèses auxquels sont appliqués des quantificateurs. Par exemple, le quantificateur suivant correspond à des chaînes du type word et word-word-word :

```
/\w+(-\w+)*/
```

Par défaut, les expressions régulières effectuent un *greedy matching*. Tout sous-modèle dans l'expression régulière (.*, par exemple) tente de mettre en correspondance autant de caractères que possible dans la chaîne avant de passer à la partie suivante de l'expression régulière. Par exemple, considérez l'expression régulière et la chaîne suivantes :

```
var pattern:RegExp = /<p>.*<\/p>/;
str:String = "<p>Paragraph 1</p> <p>Paragraph 2</p>";
```

L'expression régulière correspond à la chaîne entière :

```
<p>Paragraph 1</p> <p>Paragraph 2</p>
```

Supposez, néanmoins, que vous souhaitez établir une correspondance avec un seul groupe `<p>...</p>`. Vous pouvez procéder comme suit :

```
<p>Paragraph 1</p>
```

Ajoutez un point d'interrogation (`?`) après les quantificateurs pour qu'ils deviennent des *quantificateurs paresseux*. Par exemple, l'expression régulière suivante, qui utilise le quantificateur paresseux `*?` , correspond à `<p>` suivi du nombre minimum de caractères possible (paresseux), suivi de `</p>` :

```
/<p>.*?<\/p>/
```

Lisez attentivement les points suivants concernant les quantificateurs :

- Les quantificateurs `{0}` et `{0,0}` n'excluent pas un élément d'une correspondance.
- Ne combinez pas plusieurs quantificateurs, comme dans `/abc+*/`.
- Le caractère point (.) ne divise pas les lignes en deux à moins de définir l'indicateur `s` (`dotall`), même s'il est suivi d'un quantificateur `*`. Considérons par exemple le code qui suit :

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*<\/p>/;
trace(str.match(re)); // null;

re = /<p>.*<\/p>/s;
trace(str.match(re));
    // output: <p>Test
    //                 Multiline</p>
```

Pour plus d'informations, voir « Indicateurs et propriétés » à la page 89.

Permutation

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez le caractère `|` (barre) dans une expression régulière pour que son moteur tienne compte des autres possibilités pour une correspondance. Par exemple, l'expression régulière suivante correspond à l'un des mots `cat, dog, pig, rat` :

```
var pattern:RegExp = /cat|dog|pig|rat/;
```

Vous pouvez utiliser des parenthèses pour définir des groupes et limiter le domaine du permutateur |. L'expression régulière suivante correspond à `cat` suivi de `nap` ou `nip` :

```
var pattern:RegExp = /cat(nap|nip)/;
```

Pour plus d'informations, voir « Groupes » à la page 87.

Les deux expressions régulières suivantes (l'une utilisant le permutateur `|`, l'autre une classe de caractère (définie avec `[` et `]`)) sont équivalentes :

```
/1|3|5|7|9/
/[13579]/
```

Pour plus d'informations, voir « Classes de caractère » à la page 83.

Groupes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez spécifier un groupe dans une expression régulière en utilisant des parenthèses, comme suit :

```
/class-(\d*)/
```

Un groupe est une sous-section d'un modèle. Vous pouvez utiliser des groupes pour effectuer les opérations suivantes :

- Appliquer un quantificateur à plusieurs caractères
- Délimiter des sous-modèles à appliquer avec la permutation (à l'aide du caractère `|`)
- Capturer des correspondances de sous-chaîne (par exemple, en utilisant `\1` dans une expression régulière pour établir une correspondance avec un groupe mis en correspondance précédemment, ou en utilisant `$1` de la même façon dans la méthode `replace()` de la classe String)

Les sections suivantes fournissent des détails sur ces utilisations de groupes.

Utilisation de groupes avec des quantificateurs

Si vous n'utilisez pas de groupe, un quantificateur s'applique au caractère ou à la classe de caractère qui le précède, comme indiqué ci-dessous :

```
var pattern:RegExp = /ab*/ ;
// matches the character a followed by
// zero or more occurrences of the character b

pattern = /a\d+/;
// matches the character a followed by
// one or more digits

pattern = /a[123]{1,3}/;
// matches the character a followed by
// one to three occurrences of either 1, 2, or 3
```

Néanmoins, vous pouvez utiliser un groupe pour appliquer un quantificateur à plusieurs caractères ou classes de caractère :

```
var pattern:RegExp = /(ab)*/;
// matches zero or more occurrences of the character a
// followed by the character b, such as abababb

pattern = /(a\d)+/;
// matches one or more occurrences of the character a followed by
// a digit, such as a1a5a8a3

pattern = /(spam ){1,3}/;
// matches 1 to 3 occurrences of the word spam followed by a space
```

Pour plus d'informations sur les quantificateurs, voir « Quantificateurs » à la page 85.

Utilisation de groupes avec le permutateur (|)

Vous pouvez utiliser des groupes pour définir le groupe de caractères auquel vous souhaitez appliquer un permutateur (`|`), comme suit :

```
var pattern:RegExp = /cat|dog/;
// matches cat or dog

pattern = /ca(t|d)og/;
// matches catog or cadog
```

Utilisation de groupes pour capturer des correspondances de sous-chaîne

Lorsque vous définissez un groupe entre parenthèses standard dans un modèle, vous pouvez ensuite vous y référer dans l'expression régulière. Il s'agit d'une *backreference*. Ces types de groupes sont appelés *groupes capturés*. Par exemple, dans l'expression régulière suivante, la séquence `\1` correspond à toute sous-chaîne mise en correspondance avec le groupe entre parenthèses capturé :

```
var pattern:RegExp = /(\d+)-by-\1/;
// matches the following: 48-by-48
```

Vous pouvez spécifier jusqu'à 99 backreferences dans une expression régulière en tapant `\1,\2,...,\99`.

De même, dans la méthode `replace()` de la classe String, vous pouvez utiliser `$1–$99` pour insérer des correspondances de chaîne de groupe capturé dans la chaîne de remplacement :

```
var pattern:RegExp = /Hi, (\w+)\./;
var str:String = "Hi, Bob.";
trace(str.replace(pattern, "$1, hello."));
    // output: Bob, hello.
```

Si vous utilisez des groupes capturés, la méthode `exec()` de la classe RegExp et la méthode `match()` de la classe String renvoient des sous-chaînes qui correspondent aux groupes capturés :

```
var pattern:RegExp = /(\w+)@(\w+).(\w+)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example,com
```

Utilisation de groupes non capturés et de groupes d'anticipation

Un groupe non capturé est utilisé pour le regroupement uniquement ; il n'est pas collecté et il ne correspond pas à des backreferences numérotées. Utilisez `(?:` et `)` pour définir des groupes non capturés, comme suit :

```
var pattern = /(?:com|org|net);
```

Par exemple, notez la différence entre mettre `(com|org)` dans un groupe capturé et dans un groupe non capturé (la méthode `exec()` répertorie les groupes capturés après la correspondance complète) :

```
var pattern:RegExp = /(\w+)@(\w+).(com|org)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
// bob@example.com,bob,example,com

//noncapturing:
var pattern:RegExp = /(\w+)@(\w+).(?:com|org)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example
```

Un type spécial de groupe non capturé est le *groupe d'animation* dont il existe deux types : le *groupe d'animation positif* et le *groupe d'animation négatif.*

Utilisez `(?=` et `)` pour définir un groupe d'anticipation positif, qui spécifie que le sous-modèle dans le groupe doit établir une correspondance à la position. Néanmoins, la portion de la chaîne qui correspond au groupe d'anticipation positif peut correspondre aux modèles restants dans l'expression régulière. Par exemple, étant donné que `(?=e)` est un groupe d'anticipation positif dans le code suivant, le caractère `e` auquel il correspond peut être mis en correspondance par une partie suivante de l'expression régulière (dans ce cas, le groupe capturé, `\w*`) :

```
var pattern:RegExp = /sh(?=e)(\w*)/i;
var str:String = "Shelly sells seashells by the seashore";
trace(pattern.exec(str));
// Shelly,elly
```

Utilisez `(?=` et `)` pour définir un groupe d'anticipation négatif, qui indique que le sous-modèle dans le groupe ne doit *pas* établir une correspondance à la position. Exemple :

```
var pattern:RegExp = /sh(?!e)(\w*)/i;
var str:String = "She sells seashells by the seashore";
trace(pattern.exec(str));
// shore,ore
```

Utilisation de groupes nommés

Un groupe nommé est un type de groupe dans une expression régulière ayant un identificateur nommé. Utilisez `(?P<name>` et `)` pour définir le groupe nommé. Par exemple, l'expression régulière suivante inclut un groupe nommé avec l'identificateur nommé `digits` :

```
var pattern = /[a-z]+(?P<digits>\d+)[a-z]+/;
```

Lorsque vous utilisez la méthode `exec()`, un groupe nommé de correspondance est ajouté comme propriété du tableau `result` :

```
var myPattern:RegExp = /([a-z]+)(?P<digits>\d+)[a-z]+/;
var str:String = "a123bcd";
var result:Array = myPattern.exec(str);
trace(result.digits); // 123
```

Voici un autre exemple, qui utilise deux groupes nommés, avec les identificateurs `name` et `dom` :

```
var emailPattern:RegExp =
    /(?P<name>(\w|[_.\-])+)@(?P<dom>((\w|-)+))+\.\w{2,4}+/;
var address:String = "bob@example.com";
var result:Array = emailPattern.exec(address);
trace(result.name); // bob
trace(result.dom); // example
```

Remarque : *les groupes nommés ne font pas partie de la spécification du langage ECMAScript. Ils représentent une fonction ajoutée dans ActionScript 3.0.*

Indicateurs et propriétés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le tableau suivant répertorie les cinq indicateurs que vous pouvez définir pour des expressions régulières. Vous pouvez accéder à chaque indicateur en tant que propriété de l'objet d'expression régulière.

Indicateur	Propriété	Description
g	global	A plusieurs correspondances.
i	ignoreCase	Correspondance sensible à la casse. S'applique aux caractères A—Z et a—z mais pas à des caractères étendus tels que É et é.
m	multiline	Lorsque cet indicateur est défini, $ et ^ peuvent correspondre au début d'une ligne et à la fin d'une ligne, respectivement.
s	dotall	Lorsque cet indicateur est défini, . (point) peut correspondre au caractère de nouvelle ligne (\n).
x	extended	Autorise les expressions régulières étendues. Vous pouvez tapez des espaces dans l'expression régulière. Ils sont ignorés dans le cadre du modèle. Ceci vous permet de taper un code d'expression régulière de façon plus lisible.

Ces propriétés sont en lecture seule. Vous pouvez définir les indicateurs (g, i, m, s, x) lorsque vous définissez une variable d'expression régulière, comme suit :

```
var re:RegExp = /abc/gimsx;
```

Cependant, vous ne pouvez pas définir directement les propriétés nommées. Par exemple, le code suivant génère une erreur :

```
var re:RegExp = /abc/;
re.global = true; // This generates an error.
```

Par défaut, à moins de les spécifier dans la déclaration d'expression régulière, les indicateurs ne sont pas définis, et les propriétés correspondantes sont également définies sur `false`.

De plus, il existe deux autres propriétés d'une expression régulière :

- La propriété `lastIndex` spécifie la position d'index dans la chaîne à utiliser pour l'appel suivant à la méthode `exec()` ou `test()` d'une expression régulière.
- La propriété `source` spécifie la chaîne qui définit la portion de modèle de l'expression régulière.

L'indicateur g (global)

Lorsque l'indicateur `g` (`global`) n'est *pas* inclus, une expression régulière n'a pas plus d'une correspondance. Par exemple, avec l'indicateur `g` exclu de l'expression régulière, la méthode `String.match()` renvoie une sous-chaîne de correspondance uniquement :

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/;
trace(str.match(pattern)) // output: she
```

Lorsque l'indicateur `g` est défini, la méthode `Sting.match()` renvoie plusieurs correspondances, comme suit :

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/g;
// The same pattern, but this time the g flag IS set.
trace(str.match(pattern)); // output: she,shells,shore
```

L'indicateur i (ignoreCase)

Par défaut, les correspondances d'expression régulière sont sensibles à la casse. Lorsque vous définissez l'indicateur `i` (`ignoreCase`), le respect de la casse est ignoré. Par exemple, la lettre minuscule `s` dans l'expression régulière ne correspond pas à la lettre majuscule `S`, le premier caractère de la chaîne :

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/)); // output: 13 -- Not the first character
```

Lorsque l'indicateur `i` est défini, cependant, l'expression régulière correspond à la lettre majuscule `S` :

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/i)); // output: 0
```

L'indicateur `i` ignore le respect de la casse uniquement pour les caractères `A–Z` et `a–z`, mais par pour les caractères étendus tels que `É` et `é` .

L'indicateur m (multiline)

Si l'indicateur `m` (`multiline`) n'est pas défini, le `^` correspond au début de la chaîne et le `$` à sa fin. Si l'indicateur `m` est défini, ces caractères correspondent au début d'une ligne et à la fin d'une ligne, respectivement. Considérez la chaîne suivante, qui inclut un caractère de nouvelle ligne :

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/g)); // Match a word at the beginning of the string.
```

Même si l'indicateur `g` (`global`) est défini dans l'expression régulière, la méthode `match()` correspond à une seule sous-chaîne, car il n'existe qu'une seule correspondance pour le `^` (le début de la chaîne). Le résultat est le suivant :

```
Test
```

Voici le même code avec l'indicateur `m` défini :

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/gm)); // Match a word at the beginning of lines.
```

Cette fois, le résultat inclut les mots au début des deux lignes :

```
Test,Multiline
```

Seul le caractère `\n` signale la fin d'une ligne. Ce n'est pas le cas des caractères suivants :

- Retour de chariot (`\r`)
- Séparateur de ligne Unicode (`\u2028`)
- Séparateur de paragraphe Unicode (`\u2028`)

L'indicateur s (dotall)

Si l'indicateur `s` (`dotall` ou "dot all") n'est pas défini, un point (`.`) dans un modèle d'expression régulière ne correspond pas à un caractère de nouvelle ligne (`\n`). Par conséquent, il n'existe aucune correspondance pour l'exemple suivant :

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/;
trace(str.match(re));
```

Néanmoins, si l'indicateur `s` est défini, le point correspond au caractère de nouvelle ligne :

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/s;
trace(str.match(re));
```

Dans ce cas, la correspondance est la sous-chaîne entière dans les balises `<p>`, y compris le caractère de nouvelle ligne :

```
<p>Test
Multiline</p>
```

L'indicateur x (extended)

Les expressions régulières peuvent être difficiles à lire, notamment lorsqu'elles comprennent de nombreux métasymboles et métaséquences. Exemple :

```
/<p(>|(\s*[^>]*>)).*?<\/p>/gi
```

Lorsque vous utilisez l'indicateur `x` (`extended`) dans une expression régulière, les espaces vides que vous tapez dans le modèle sont ignorés. Par exemple, l'expression régulière suivante est identique à l'exemple précédent :

```
/     <p     (>   |   (\s*   [^>]*   >))     .*?     <\/p>   /gix
```

Si l'indicateur `x` est défini et que vous souhaitez établir une correspondance avec un espace vide, faites précéder l'espace vide d'une barre oblique. Par exemple, les deux expressions régulières suivantes sont équivalentes :

```
/foo bar/
/foo \ bar/x
```

La propriété lastIndex

La propriété `lastIndex` spécifie la position d'index dans la chaîne à laquelle la recherche suivante doit démarrer. Cette propriété affecte les méthodes `exec()` et `test()` appelées sur une expression régulière dont l'indicateur `g` est défini sur `true`. Considérons par exemple le code qui suit :

```
var pattern:RegExp = /p\w*/gi;
var str:String = "Pedro Piper picked a peck of pickled peppers.";
trace(pattern.lastIndex);
var result:Object = pattern.exec(str);
while (result != null)
{
    trace(pattern.lastIndex);
    result = pattern.exec(str);
}
```

La propriété `lastIndex` est défini sur 0 par défaut (pour commencer les recherches au début de la chaîne). Après chaque correspondance, elle est définie sur la position d'index suivant la correspondance. Par conséquent, le résultat pour le code précédent est le suivant :

```
0
5
11
18
25
36
44
```

Si l'indicateur `global` est défini sur `false`, les méthodes `exec()` et `test()` n'utilisent pas ni ne définissent la propriété `lastIndex`.

Les méthodes `match()`, `replace()` et `search()` de la classe String lancent toutes les recherches du début de la chaîne, indépendamment du réglage de la propriété `lastIndex` de l'expression régulière utilisée dans l'appel à la méthode (néanmoins, la méthode `match()` définit `lastIndex` sur 0).

Vous pouvez définir la propriété `lastIndex` de sorte à ajuster la position de début dans la chaîne pour la mise en correspondance des expressions régulières.

La propriété source

La propriété `source` spécifie la chaîne qui définit la portion de modèle d'une expression régulière. Exemple :

```
var pattern:RegExp = /foo/gi;
trace(pattern.source); // foo
```

Méthodes d'utilisation d'expressions régulières avec des chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe RegExp comprend deux méthodes : `exec()` et `test()`.

Outre les méthodes `exec()` et `test()` de la classe RegExp, la classe String comprend les méthodes suivantes qui vous permettent d'établir une correspondance avec des expressions régulières dans des chaînes : `match()`, `replace()`, `search()` et `splice()`.

La méthode test()

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `test()` de la classe RegExp vérifie si la chaîne fournie contient une correspondance pour l'expression régulière, comme l'indique l'exemple suivant :

```
var pattern:RegExp = /Class-\w/;
var str = "Class-A";
trace(pattern.test(str)); // output: true
```

La méthode exec()

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `exec()` de la classe RegExp vérifie si la chaîne fournie contient une correspondance pour l'expression régulière et renvoie un tableau avec les éléments suivants :

- La sous-chaîne correspondante
- Les correspondances de sous-chaîne pour les groupes entre parenthèses dans l'expression régulière

Le tableau inclut également une propriété `index` qui indique la position d'index du début de la correspondance de sous-chaîne.

Considérons par exemple le code qui suit :

```
var pattern:RegExp = /\d{3}\-\d{3}-\d{4}/; //U.S phone number
var str:String = "phone: 415-555-1212";
var result:Array = pattern.exec(str);
trace(result.index, " - ", result);
// 7-415-555-1212
```

Utilisez la méthode `exec()` plusieurs fois pour établir une correspondance avec plusieurs sous-chaînes lorsque l'indicateur `g` (`global`) est défini pour l'expression régulière :

```
var pattern:RegExp = /\w*sh\w*/gi;
var str:String = "She sells seashells by the seashore";
var result:Array = pattern.exec(str);

while (result != null)
{
    trace(result.index, "\t", pattern.lastIndex, "\t", result);
    result = pattern.exec(str);
}
//output:
// 0  3  She
// 10  19  seashells
// 27  35  seashore
```

Méthodes de chaîne utilisant des paramètres RegExp

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes suivantes de la classe String prennent des expressions régulières comme paramètres : `match()`, `replace()`, `search()` et `split()`. Pour plus d'informations sur ces méthodes, voir « Recherche de modèles dans des chaînes et remplacement de sous-chaînes » à la page 17.

Exemple d'expression régulière : analyseur Wiki

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple simple de conversion de texte Wiki illustre des utilisations d'expressions régulières :

- Conversion de lignes de texte qui mettent en correspondance un modèle Wiki source et les chaînes de sortie HTML appropriées.
- Utilisation d'une expression régulière pour convertir des modèles URL en balises de lien hypertexte HTML `<a>`
- Utilisation d'une expression régulière pour convertir les chaînes dollar américain (`"$9.95"`, par exemple) en chaînes euro (`"8.24 €"`, par exemple)

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application WikiEditor se trouvent dans le dossier Samples/WikiEditor. L'application se compose des fichiers suivants :

Fichier	Description
WikiEditor.mxml ou WikiEditor.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).

Fichier	Description
com/example/programmingas3/regExpExamples/WikiParser.as	Une classe qui comprend des méthodes utilisant des expressions régulières pour convertir des modèles de texte d'entrée Wiki en sortie HTML équivalente.
com/example/programmingas3/regExpExamples/URLParser.as	Une classe qui comprend des méthodes utilisant des expressions régulières pour convertir des chaînes URL en balises de lien hypertexte HTML `<a>`.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	Une classe qui comprend des méthodes utilisant des expressions régulières pour convertir des chaînes dollar américain en chaînes euro.

Définition de la classe WikiParser

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe WikiParser inclut des méthodes qui convertissent le texte d'entrée Wiki en sortie HTML équivalente. Il ne s'agit pas d'une application de conversion Wiki très puissante, mais elle illustre des utilisations d'expressions régulières pour la mise en correspondance de modèle et la conversion de chaîne.

La fonction constructeur et la méthode `setWikiData()` initialisent un exemple de chaîne de texte d'entrée Wiki, comme suit :

```
public function WikiParser()
{
    wikiData = setWikiData();
}
```

Lorsque l'utilisateur clique sur le bouton de test dans l'application exemple, cette dernière appelle la méthode `parseWikiString()` de l'objet WikiParser. Cette méthode appelle plusieurs autres méthodes, qui assemblent à leur tour la chaîne HTML résultante.

```
public function parseWikiString(wikiString:String):String
{
    var result:String = parseBold(wikiString);
    result = parseItalic(result);
    result = linesToParagraphs(result);
    result = parseBullets(result);
    return result;
}
```

Chaque méthode appelée—`parseBold()`, `parseItalic()`, `linesToParagraphs()`, et `parseBullets()`—utilise la méthode `replace()` de la chaîne pour remplacer les modèles de correspondance, définis par une expression régulière, de façon à transformer le texte Wiki en texte formaté HTML.

Conversion des modèles de texte en gras et en italique

La méthode `parseBold()` recherche un modèle de texte Wiki en gras (`'''foo'''`, par exemple) et le transforme en son équivalent HTML (`<b>foo</b>`, par exemple), comme suit :

```
private function parseBold(input:String):String
{
    var pattern:RegExp = /'''(.*?)'''/g;
    return input.replace(pattern, "<b>$1</b>");
}
```

Notez que la portion `(.?*)` de l'expression régulière correspond à des caractères (`*`) entre les deux modèles de définition `'''`. Le quantificateur `?` rend la correspondance nongreedy, de façon à ce que pour une chaîne telle que `'''aaa''' bbb '''ccc'''`, la première chaîne mise en correspondance soit `'''aaa'''` et non la chaîne entière (qui commence et se termine par le modèle `'''`).

Les parenthèses dans l'expression régulière définissent un groupe capturé, et la méthode `replace()` se réfère à ce groupe en utilisant le code `$1` dans la chaîne de remplacement. L'indicateur `g` (`global`) dans l'expression régulière vérifie que la méthode `replace()` remplace toutes les correspondances dans la chaîne (pas simplement la première).

La méthode `parseItalic()` fonctionne comme la méthode `parseBold()`, sauf qu'elle recherche deux apostrophes (`''`) comme séparateur pour le texte en italique (au lieu de trois) :

```
private function parseItalic(input:String):String
{
    var pattern:RegExp = /''(.*?)''/g;
    return input.replace(pattern, "<i>$1</i>");
}
```

Conversion de modèles de puce

Comme dans l'exemple suivant, la méthode `parseBullet()` recherche le modèle de puce Wiki (`* foo`, par exemple) et le transforme en son équivalent HTML (`<li>foo</li>`, par exemple) :

```
private function parseBullets(input:String):String
{
    var pattern:RegExp = /^\*(.*)/gm;
    return input.replace(pattern, "<li>$1</li>");
}
```

Le symbole `^` au début de l'expression régulière correspond au début d'une ligne. L'indicateur `m` (`multiline`) dans l'expression régulière fait en sorte que cette dernière compare le symbole `^` au début d'une ligne, et non simplement au début de la chaîne.

Le modèle `\*` correspond à un astérisque (la barre oblique est utilisée pour signaler un astérisque littéral au lieu d'un quantificateur `*`).

Les parenthèses dans l'expression régulière définissent un groupe capturé, et la méthode `replace()` se réfère à ce groupe en utilisant le code `$1` dans la chaîne de remplacement. L'indicateur `g` (`global`) dans l'expression régulière vérifie que la méthode `replace()` remplace toutes les correspondances dans la chaîne (pas simplement la première).

Conversion de modèles Wiki de paragraphe

La méthode `linesToParagraphs()` convertit chaque ligne dans la chaîne Wiki d'entrée en une balise de paragraphe HTML `<p>`. Ces lignes dans la méthode extraient des lignes vides de la chaîne d'entrée Wiki :

```
var pattern:RegExp = /^$/gm;
var result:String = input.replace(pattern, "");
```

Les symboles `^` et `$` dans l'expression régulière correspondent au début et à la fin d'une ligne. L'indicateur `m` (`multiline`) dans l'expression régulière fait en sorte que cette dernière compare le symbole ^ au début d'une ligne, et non simplement au début de la chaîne.

La méthode `replace()` remplace toutes les chaînes de correspondance (lignes vides) par une chaîne vide (`""`). L'indicateur `g` (`global`) dans l'expression régulière vérifie que la méthode `replace()` remplace toutes les correspondances dans la chaîne (pas simplement la première).

Conversion d'URL en balises HTML <a>

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur clique sur le bouton de test dans l'exemple d'application et qu'il a coché la case `urlToATag`, l'application appelle la méthode statique `URLParser.urlToATag()` pour convertir les chaînes URL de la chaîne Wiki d'entrée en balises HTML `<a>`.

```
var protocol:String = "((?:http|ftp)://)";
var urlPart:String = "([a-z0-9_-]+\.[a-z0-9_-]+)";
var optionalUrlPart:String = "(\.[a-z0-9_-]*)";
var urlPattern:RegExp = new RegExp(protocol + urlPart + optionalUrlPart, "ig");
var result:String = input.replace(urlPattern, "<a href='$1$2$3'><u>$1$2$3</u></a>");
```

La fonction constructeur `RegExp()` sert à assembler une expression régulière (`urlPattern`) à partir de plusieurs parties constituantes. Ces parties constituantes sont représentées par chaque chaîne définissant une partie du modèle de l'expression régulière.

La première partie du modèle de l'expression régulière, définie par la chaîne `protocol`, définit un protocole URL : `http://` ou `ftp://`. Les parenthèses définissent un groupe non capturé, indiqué par le symbole `?`. Ceci signifie que les parenthèses servent simplement à définir un groupe pour le modèle de permutation `|` ; le groupe ne correspondra pas à des codes de backreference (`$1`, `$2`, `$3`) dans la chaîne de remplacement de la méthode `replace()`.

Les autres parties constituantes de l'expression régulière utilisent chacune des groupes capturés (indiqués par des parenthèses dans le modèle) qui sont ensuite utilisés dans les codes de backreference (`$1`, `$2`, `$3`) dans la chaîne de remplacement de la méthode `replace()`.

La partie du modèle définie par la chaîne `urlPart` correspond *au moins* à l'un des caractères suivants : `a-z`, `0-9`, _ ou `-`. La quantificateur `+` indique qu'au moins un caractère a une correspondance. Le `\.` indique un point obligatoire (`.`). Et le reste correspond à une autre chaîne d'au moins un de ces caractères : `a-z`, `0-9`, _ ou `-`.

La partie du modèle définie par la chaîne `optionalUrlPart` correspond à *zéro ou plus* des caractères suivants : un point (`.`) suivi par n'importe quel nombre de caractères alphanumériques (y compris _ et `-`). Le quantificateur `*` indique que zéro ou plus de caractères ont une correspondance.

L'appel à la méthode `replace()` utilise l'expression régulière et assemble la chaîne HTML de remplacement, à l'aide de backreferences.

La méthode `urlToATag()` appelle ensuite la méthode `emailToATag()`, qui utilise des techniques semblables pour remplacer des modèles de courrier électronique par des chaînes de lien hypertexte HTML `<a>`. Les expressions régulières utilisées pour correspondre à des URL HTTP, FTP et de courrier électronique dans le fichier sont assez simples car elles sont données à titre d'exemple. Elles sont beaucoup plus compliquées si l'on souhaite établir une correspondance plus correcte.

Conversion des chaînes dollar américain en chaînes euro

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur clique sur le bouton de test dans l'exemple d'application et qu'il a coché la case `dollarToEuro` , l'application appelle la méthode statique `CurrencyConverter.usdToEuro()` pour convertir des chaînes dollar américain (`"$9.95"`, par exemple) en chaînes euro (`"8.24€"`, par exemple), comme suit :

```
var usdPrice:RegExp = /\$([\d,]+.\d+)+/g;
return input.replace(usdPrice, usdStrToEuroStr);
```

La première ligne définit un modèle simple pour établir une correspondance avec des chaînes dollar américain. Le caractère `$` est précédé du caractère d'échappement (`\`).

La méthode `replace()` utilise l'expression régulière pour établir une correspondance avec le modèle et appelle la fonction `usdStrToEuroStr()` pour déterminer la chaîne de remplacement (une valeur en euros).

Lorsqu'un nom de fonction est utilisé comme deuxième paramètre de la méthode `replace()`, les éléments suivants sont transmis comme paramètres à la fonction appelée :

- La partie correspondante de la chaîne.
- Tout groupe entre parenthèses capturé correspondant. Le nombre d'arguments transmis de cette façon varie selon le nombre de correspondances de groupes entre parenthèses capturées. Pour déterminer le nombre de correspondances de groupes entre parenthèses capturés, vérifiez `arguments.length - 3` dans le code de la fonction.
- La position d'index dans la chaîne où débute la correspondance.
- La chaîne complète.

La méthode `usdStrToEuroStr()` convertit les modèles de chaîne dollar américain en chaînes euro, comme suit :

```
private function usdToEuro(...args):String
{
    var usd:String = args[1];
    usd = usd.replace(",", "");
    var exchangeRate:Number = 0.828017;
    var euro:Number = Number(usd) * exchangeRate;
    trace(usd, Number(usd), euro);
    const euroSymbol:String = String.fromCharCode(8364); // €
    return euro.toFixed(2) + " " + euroSymbol;
}
```

`args[1]` représente le groupe entre parenthèses capturé mis en correspondance par l'expression régulière `usdPrice`. Il s'agit de la portion numérique de la chaîne dollar américain, c'est-à-dire la quantité en dollar, sans le signe `$`. La méthode applique une conversion de taux de change et renvoie la chaîne résultante (avec un symbole de fin de ligne € au lieu du symbole $ placé à gauche).

Chapitre 6 : Utilisation de XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 inclut un groupe de classes basé sur la spécification ECMAScript pour XML (E4X) (ECMA-357 version 2). Ces classes comprennent une fonctionnalité puissante et facile à utiliser permettant de travailler avec des données XML. A l'aide d'E4X, vous pouvez développer un code avec des données XML plus rapidement qu'avec les techniques de programmation précédentes. En outre, le code que vous produisez est plus facile à lire.

Voir aussi

Classe XML

Spécification ECMA-357

Principes de base de XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

XML est un moyen standard de représenter des informations structurées afin de permettre aux ordinateurs de les utiliser facilement et aux utilisateurs de les écrire et de les comprendre. XML est l'abréviation de eXtensible Markup Language. La norme XML est disponible à l'adresse www.w3.org/XML/.

XML offre une façon pratique et standard de classer des données, afin de faciliter leur lecture, leur accès et leur manipulation. XML utilise une arborescence et une structure de balises identiques à celle du code HTML. Voici un exemple simple de données XML :

```
<song>
    <title>What you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

Les données XML peuvent également être plus complexes, avec des balises imbriquées dans d'autres balises ainsi que des attributs et d'autres composants structurels. Voici un exemple plus complexe de données XML :

```
<album>
    <title>Questions, unanswered</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <tracks>
        <song tracknumber="1" length="4:05">
            <title>What do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:31</lastplayed>
        </song>
        <song tracknumber="2" length="3:45">
            <title>Who do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:35</lastplayed>
        </song>
        <song tracknumber="3" length="5:14">
            <title>When do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:39</lastplayed>
        </song>
        <song tracknumber="4" length="4:19">
            <title>Do you know?</title>
            <artist>Steve and the flubberblubs</artist>
            <lastplayed>2006-10-17-08:44</lastplayed>
        </song>
    </tracks>
</album>
```

Vous remarquerez que ce document XML contient d'autres structures XML complètes (les balises `song` avec leurs enfants, par exemple). Il démontre également d'autres structures XML telles que des attributs (`tracknumber` et `length` dans les balises `song`), et des balises qui contiennent d'autres balises au lieu de données (la balise `tracks`, par exemple).

Prise en main de XML

Si vous avez peu ou pas d'expérience avec XML, voici une brève description des aspects les plus courants des données XML. Les données XML sont écrites sous forme de texte brut, avec une syntaxe spécifique permettant d'organiser les informations en un format structuré. Généralement, un seul jeu de données XML est appelé un *document XML*. Dans le format XML, les données sont organisées en *éléments* (qui peuvent être des éléments de données uniques ou des conteneurs pour d'autres éléments) à l'aide d'une structure hiérarchique. Chaque document XML possède un élément comme niveau supérieur ou élément principal. Cet élément racine peut contenir une seule information, même s'il est plus probable qu'il contienne d'autres éléments qui, à leur tour, contiennent d'autres éléments, et ainsi de suite. Par exemple, ce document XML contient les informations relatives à un album musical :

```
<song tracknumber="1" length="4:05">
    <title>What do you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <mood>Happy</mood>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

Chaque élément se distingue par un ensemble de *balises*—le nom de l'élément placé entre chevrons (signes inférieur à et supérieur à). La balise de début, indiquant le début de l'élément, porte le nom de l'élément :

```
<title>
```

La balise de fin, qui marque la fin de l'élément, a une barre oblique avant le nom de l'élément :

```
</title>
```

Si un élément est vide, il peut être écrit comme élément vide (parfois appelé élément de fin automatique). Dans XML, cet élément :

```
<lastplayed/>
```

est identique à cet élément :

```
<lastplayed></lastplayed>
```

Outre le contenu de l'élément placé entre les balises de début et de fin, un élément peut comporter d'autres valeurs, appelées *attributs*, définies dans la balise de début de l'élément. Par exemple, cet élément XML définit un seul attribut appelé `length`, avec la valeur `"4:19"` :

```
<song length="4:19"></song>
```

Chaque élément XML possède un contenu qui est soit une valeur, soit un ou plusieurs éléments XML, ou rien du tout (pour un élément vide).

En savoir plus sur XML

Pour en savoir plus sur l'utilisation de XML, un grand nombre de livres et de ressources sont disponibles, ainsi que les sites Web suivants :

- Didacticiel XML W3Schools : http://w3schools.com/xml/
- Didacticiels XMLpitstop, listes de discussions, etc. : http://xmlpitstop.com/

Classes ActionScript pour utiliser XML

ActionScript 3.0 inclut plusieurs classes permettant d'utiliser des informations structurées XML. Les deux classes principales sont les suivantes :

- XML : représente un seul élément XML, qui peut être un document XML avec plusieurs enfants ou un élément à une seule valeur dans un document.
- XMLList : représente un ensemble d'éléments XML. Un objet XMLList est utilisé lorsque plusieurs éléments XML sont des frères (au même niveau, et contenus par le même parent, dans la hiérarchie du document XML). Par exemple, une occurrence de XMLList serait le moyen le plus facile d'utiliser cet ensemble d'éléments XML (probablement contenus dans un document XML):

```
<artist type="composer">Fred Wilson</artist>
<artist type="conductor">James Schmidt</artist>
<artist type="soloist">Susan Harriet Thurndon</artist>
```

Pour des utilisations plus avancées impliquant des espaces de noms XML, ActionScript inclut également les classes Namespace et QName. Pour plus d'informations, voir « Utilisation des espaces de noms XML » à la page 114.

Outre les classes intégrées permettant d'utiliser XML, ActionScript 3.0 comprend également plusieurs opérateurs offrant des fonctionnalités spécifiques pour accéder et manipuler des données XML. Cette approche d'utilisation de XML au moyen de ces classes et de ces opérateurs est appelée ECMAScript pour XML (E4X), comme défini par la spécification ECMA-357 version 2.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants utilisés dans le cadre de la programmation de routines de gestion des éléments XML :

Elément Entité unique dans un document XML correspondant au contenu compris entre une balise de début et une balise de fin (balises comprises). Les éléments XML peuvent contenir des données de texte ou d'autres éléments, ou bien être vides.

Elément vide Elément XML qui ne contient aucun élément enfant. Les éléments vides sont souvent écrits en tant que balises de fin automatique (`<element/>`, par exemple).

Document Structure XML unique. Un document XML peut contenir n'importe quel nombre d'éléments (ou peut être constitué d'un seul élément vide) ; cependant, un document XML doit avoir un seul élément de niveau supérieur qui contient tous les autres éléments dans le document.

Nœud Autre nom d'un élément XML.

Attribut Valeur nommée associée à un élément qui est écrite dans la balise de début de l'élément au format `attributename="valeur"`, plutôt que d'être écrite comme élément enfant séparé imbriqué dans l'élément.

Approche E4X concernant le traitement XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La spécification ECMAScript pour XML définit un ensemble de classes et de fonctionnalités permettant d'utiliser des données XML. Cet ensemble de classes et de fonctionnalités est connu sous le nom de *E4X*. ActionScript 3.0 intègre les classes E4X suivantes : XML, XMLList, QName et Namespace.

Les méthodes, propriétés et opérateurs des classes E4X sont conçus avec les objectifs suivants :

- Simplicité : lorsque cela est possible, E4X facilite l'écriture et la compréhension du code à utiliser avec des données XML.
- Cohérence : les méthodes et le raisonnement E4X sont cohérents avec eux-mêmes et avec d'autres parties d'ActionScript.
- Connaissance : vous manipulez des données XML avec des opérateurs bien connus tels que l'opérateur point (.).

Remarque : *ActionScript 2.0 intègre une autre classe XML. Dans ActionScript 3.0, cette classe a été renommée XMLDocument, pour éviter tout conflit de nom avec la classe XML d'ActionScript 3.0 qui fait partie d'E4X. Dans ActionScript 3.0, les classes héritées (XMLDocument, XMLNode, XMLParser et XMLTag) sont comprises dans le package flash.xml, pour la prise en charge du contenu hérité principalement. Les nouvelles classes E4X sont des classes de base. Vous ne devez pas importer de package pour les utiliser. Pour plus d'informations sur les classes XML d'ActionScript 2.0 existantes, voir le package flash.xml dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.*

Voici un exemple de manipulation de données avec E4X :

```
var myXML:XML =
    <order>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Votre application charge souvent des données XML depuis une source externe telle qu'un service Web ou un flux RSS. Toutefois, par souci de clarté, les exemples de code ci-après affectent les données XML en tant que littéraux.

Comme l'indique le code suivant, E4X inclut des opérateurs intuitifs tels que les opérateurs point (.) et identifiant d'attribut (@) pour accéder aux propriétés et aux attributs dans l'XML :

```
trace(myXML.item[0].menuName); // Output: burger
trace(myXML.item.(@id==2).menuName); // Output: fries
trace(myXML.item.(menuName=="burger").price); // Output: 3.95
```

Utilisez la méthode `appendChild()` pour affecter un nouveau nœud enfant à l'XML, comme l'indique le code suivant :

```
var newItem:XML =
    <item id="3">
        <menuName>medium cola</menuName>
        <price>1.25</price>
    </item>

myXML.appendChild(newItem);
```

Utilisez les opérateurs `@` et `.` non seulement pour lire des données mais également pour les affecter, comme dans l'exemple suivant :

```
myXML.item[0].menuName="regular burger";
myXML.item[1].menuName="small fries";
myXML.item[2].menuName="medium cola";

myXML.item.(menuName=="regular burger").@quantity = "2";
myXML.item.(menuName=="small fries").@quantity = "2";
myXML.item.(menuName=="medium cola").@quantity = "2";
```

Utilisez une boucle `for` pour parcourir en boucle les nœuds de l'XML, comme suit :

```
var total:Number = 0;
for each (var property:XML in myXML.item)
{
    var q:int = Number(property.@quantity);
    var p:Number = Number(property.price);
    var itemTotal:Number = q * p;
    total += itemTotal;
    trace(q + " " + property.menuName + " $" + itemTotal.toFixed(2))
}
trace("Total: $", total.toFixed(2));
```

Objets XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet XML peut représenter un élément XML, un attribut, un commentaire, une instruction de traitement ou un élément de texte.

Un objet XML est classé soit dans la catégorie des objets ayant un *contenu simple*, soit dans celle des objets ayant un *contenu complexe*. Un objet XML ayant des nœuds enfant est considéré comme ayant un contenu complexe. Le contenu est considéré comme simple s'il correspond à un attribut, un commentaire, une instruction de traitement ou un nœud de texte.

Par exemple, l'objet XML suivant contient un contenu complexe, y compris un commentaire et une instruction de traitement :

```
XML.ignoreComments = false;
XML.ignoreProcessingInstructions = false;
var x1:XML =
    <order>
        <!--This is a comment. -->
        <?PROC_INSTR sample ?>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Comme l'indique l'exemple suivant, vous pouvez maintenant utiliser les méthodes `comments()` et `processingInstructions()` pour créer des objets XML, un commentaire et une instruction de traitement :

```
var x2:XML = x1.comments()[0];
var x3:XML = x1.processingInstructions()[0];
```

Propriétés XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe XML a cinq propriétés statiques :

- Les propriétés `ignoreComments` et `ignoreProcessingInstructions` déterminent si les commentaires ou les instructions de traitement sont ignorés lorsque l'objet XML est analysé.
- La propriété `ignoreWhitespace` détermine si les espaces blancs sont ignorés dans les balises d'un élément et les expressions intégrées séparées uniquement par des espaces blancs.
- Les propriétés `prettyIndent`et `prettyPrinting` servent à formater le texte renvoyé par les méthodes `toString()` et `toXMLString()` de la classe XML.

Pour plus d'informations sur ces propriétés, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Méthodes XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes suivantes vous permettent d'utiliser la structure hiérarchique des objets XML :

- `appendChild()`
- `child()`
- `childIndex()`
- `children()`
- `descendants()`
- `elements()`
- `insertChildAfter()`

- `insertChildBefore()`
- `parent()`
- `prependChild()`

Les méthodes suivantes vous permettent d'utiliser des attributs d'objet XML :

- `attribute()`
- `attributes()`

Les méthodes suivantes vous permettent d'utiliser des propriétés d'objet XML :

- `hasOwnProperty()`
- `propertyIsEnumerable()`
- `replace()`
- `setChildren()`

Les méthodes suivantes vous permettent d'utiliser des espaces de nom et des noms qualifiés :

- `addNamespace()`
- `inScopeNamespaces()`
- `localName()`
- `name()`
- `namespace()`
- `namespaceDeclarations()`
- `removeNamespace()`
- `setLocalName()`
- `setName()`
- `setNamespace()`

Les méthodes suivantes vous permettent d'utiliser et de déterminer certains types de contenu XML :

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `nodeKind()`
- `processingInstructions()`
- `text()`

Les méthodes suivantes vous permettent d'effectuer des conversions en chaînes et de formater des objets XML :

- `defaultSettings()`
- `setSettings()`
- `settings()`
- `normalize()`
- `toString()`
- `toXMLString()`

Il existe quelques méthodes supplémentaires :

- `contains()`
- `copy()`
- `valueOf()`
- `length()`

Pour plus d'informations sur ces méthodes, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Objets XMLList

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une occurrence de XMLList représente un ensemble arbitraire d'objets XML. Elle peut contenir des documents XML complets, des fragments XML ou les résultats d'une requête XML.

Les méthodes suivantes vous permettent d'utiliser la structure hiérarchique des objets XMLList :

- `child()`
- `children()`
- `descendants()`
- `elements()`
- `parent()`

Les méthodes suivantes vous permettent d'utiliser des attributs d'objet XMLList :

- `attribute()`
- `attributes()`

Les méthodes suivantes vous permettent d'utiliser des propriétés XMLList :

- `hasOwnProperty()`
- `propertyIsEnumerable()`

Les méthodes suivantes vous permettent d'utiliser et de déterminer certains types de contenu XML :

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `processingInstructions()`
- `text()`

Les méthodes suivantes vous permettent d'effectuer des conversions en chaînes et de formater les objets XMLList :

- `normalize()`
- `toString()`
- `toXMLString()`

Il existe quelques méthodes supplémentaires :

- `contains()`
- `copy()`
- `length()`
- `valueOf()`

Pour plus d'informations sur ces méthodes, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Pour un objet XMLList qui contient exactement un élément XML, vous pouvez utiliser toutes les propriétés et les méthodes de la classe XML car un XMLList avec un élément XML est traité comme un objet XML. Par exemple, dans le code suivant, étant donné que `doc.div` est un objet XMLList contenant un élément, vous pouvez utiliser la méthode `appendChild()` de la classe XML :

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

Pour consulter la liste des méthodes et des propriétés XML, voir « Objets XML » à la page 103.

Initialisation de variables XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez affecter un littéral XML à un objet XML, comme suit :

```
var myXML:XML =
    <order>
        <item id='1'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>
```

Comme indiqué dans le code suivant, vous pouvez également utiliser le constructeur `new` pour créer une occurrence d'un objet XML à partir d'une chaîne contenant des données XML :

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                            + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

Si les données XML dans la chaîne ne sont pas bien formées (par exemple, s'il manque une balise de fin), une erreur d'exécution se produit.

Vous pouvez également transmettre des données par référence (à partir d'autres variables) dans un objet XML, comme indiqué dans l'exemple suivant :

```
var tagname:String = "item";
var attributename:String = "id";
var attributevalue:String = "5";
var content:String = "Chicken";
var x:XML = <{tagname} {attributename}={attributevalue}>{content}</{tagname}>;
trace(x.toXMLString())
    // Output: <item id="5">Chicken</item>
```

Pour charger des données XML depuis une URL, utilisez la classe URLLoader, comme indiqué dans l'exemple suivant :

```
import flash.events.Event;
import flash.net.URLLoader;
import flash.net.URLRequest;

var externalXML:XML;
var loader:URLLoader = new URLLoader();
var request:URLRequest = new URLRequest("xmlFile.xml");
loader.load(request);
loader.addEventListener(Event.COMPLETE, onComplete);

function onComplete(event:Event):void
{
    var loader:URLLoader = event.target as URLLoader;
    if (loader != null)
    {
        externalXML = new XML(loader.data);
        trace(externalXML.toXMLString());
    }
    else
    {
        trace("loader is not a URLLoader!");
    }
}
```

Pour lire des données XML depuis une connexion de socket, utilisez la classe XMLSocket . Pour plus d'informations, voir la classe XMLSocket dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Assemblage et transformation d'objets XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez la méthode `prependChild()` ou `appendChild()` pour ajouter une propriété au début ou à la fin de la liste des propriétés d'un objet XML, comme indiqué dans l'exemple suivant :

```
var x1:XML = <p>Line 1</p>
var x2:XML = <p>Line 2</p>
var x:XML = <body></body>
x = x.appendChild(x1);
x = x.appendChild(x2);
x = x.prependChild(<p>Line 0</p>);
    // x == <body><p>Line 0</p><p>Line 1</p><p>Line 2</p></body>
```

Utilisez la méthode `insertChildBefore()` ou `insertChildAfter()` pour ajouter une propriété avant ou après une propriété spécifiée, comme suit :

```
var x:XML =
    <body>
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
    </body>
var newNode:XML = <p>Paragraph 1.5</p>
x = x.insertChildAfter(x.p[0], newNode)
x = x.insertChildBefore(x.p[2], <p>Paragraph 1.75</p>)
```

Comme indiqué dans l'exemple suivant, vous pouvez également utiliser des accolades ({ et }) pour transmettre des données par référence (à partir d'autres variables) lorsque vous construisez des objets XML :

```
var ids:Array = [121, 122, 123];
var names:Array = [["Murphy","Pat"], ["Thibaut","Jean"], ["Smith","Vijay"]]
var x:XML = new XML("<employeeList></employeeList>");

for (var i:int = 0; i < 3; i++)
{
    var newnode:XML = new XML();
    newnode =
        <employee id={ids[i]}>
            <last>{names[i][0]}</last>
            <first>{names[i][1]}</first>
        </employee>;

    x = x.appendChild(newnode)
}
```

Vous pouvez affecter des propriétés et des attributs à un objet XML à l'aide de l'opérateur =, comme dans l'exemple suivant :

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

Ceci définit l'objet XML `x` de la façon suivante :

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

Vous pouvez utiliser les opérateurs + et += pour concaténer des objets XMLList :

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

Ceci définit l'objet XMLList `xList` de la façon suivante :

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

Parcours de structures XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'une des fonctions puissantes d'XML est sa capacité à fournir des données imbriquées, complexes, via une chaîne linéaire de caractères de texte. Lorsque vous chargez des données dans un objet XML, ActionScript les analyse et charge sa structure hiérarchique en mémoire (ou envoie une erreur d'exécution si les données XML ne sont pas formées correctement).

Les opérateurs et les méthodes des objets XML et XMLList permettent de parcourir aisément la structure des données XML.

Utilisez l'opérateur point (.) et l'opérateur d'accesseur descendant (..) pour accéder aux propriétés enfant d'un objet XML. Considérez l'objet XML suivant :

```
var myXML:XML =
    <order>
        <book ISBN="0942407296">
            <title>Baking Extravagant Pastries with Kumquats</title>
            <author>
                <lastName>Contino</lastName>
                <firstName>Chuck</firstName>
            </author>
            <pageCount>238</pageCount>
        </book>
        <book ISBN="0865436401">
            <title>Emu Care and Breeding</title>
            <editor>
                <lastName>Case</lastName>
                <firstName>Justin</firstName>
            </editor>
            <pageCount>115</pageCount>
        </book>
    </order>
```

L'objet `myXML.book` est un objet XMLList contenant des propriétés enfant de l'objet `myXML` appelées `book`. Ces deux objets XML correspondent aux deux propriétés `book` de l'objet `myXML`.

L'objet `myXML..lastName` est un objet XMLList contenant des propriétés descendantes appelées `lastName`. Ces deux objets XML correspondent aux deux propriétés `lastName` de l'objet `myXML`.

L'objet `myXML.book.editor.lastName` est un objet XMLList contenant tout enfant appelé `lastName` des enfants appelés `editor` des enfants appelés `book` de l'objet `myXML` : en l'occurrence, un objet XMLList contenant un seul objet XML (la propriété `lastName` dont la valeur correspond à « `Case` »).

Accès aux nœuds enfant et parent

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `parent()` renvoie le parent d'un objet XML.

Vous pouvez utiliser les valeurs d'index ordinales d'une liste enfant pour accéder à des objets enfant spécifiques. Par exemple, considérez un objet XML `myXML` ayant deux propriétés enfant appelées `book`. Chaque propriété enfant appelée `book` possède un numéro d'index qui lui est associé :

```
myXML.book[0]
myXML.book[1]
```

Pour accéder à un petit-enfant spécifique, vous pouvez indiquer des numéros d'index pour les noms de l'enfant et du petit-enfant :

```
myXML.book[0].title[0]
```

Cependant, s'il n'existe qu'un seul enfant de `x.book[0]` nommé `title`, vous pouvez omettre la référence d'index, comme suit :

```
myXML.book[0].title
```

De même, s'il n'existe qu'un seul enfant book de l'objet `x` et que cet objet enfant possède un seul objet title, vous pouvez omettre les deux références d'index, de la façon suivante :

```
myXML.book.title
```

Vous pouvez utiliser la méthode `child()` pour accéder à des enfants dont le nom est basé sur une variable ou une expression, comme indiqué dans l'exemple suivant :

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

Accès à des attributs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez le symbole `@` (l'opérateur identifiant d'attribut) pour accéder aux attributs dans un objet XML ou XMLList, comme indiqué dans le code suivant :

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

Vous pouvez utiliser le symbole de caractère générique `*` avec le symbole `@` pour accéder à tous les attributs d'un objet XML ou XMLList, comme dans le code suivant :

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

Vous pouvez utiliser la méthode `attribute()` ou `attributes()` pour accéder à un attribut spécifique ou à tous les attributs d'un objet XML ou XMLList, comme dans le code suivant :

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.attribute("id")); // 6401
trace(employee.attribute("*").toXMLString());
// 6401
// 233
trace(employee.attributes().toXMLString());
// 6401
// 233
```

Vous pouvez également utiliser la syntaxe suivante pour accéder à des attributs, comme indiqué dans l'exemple suivant :

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

Ils sont tous équivalents à `employee.@id`. Cependant, la syntaxe `employee.@id` est préférable.

Filtrage par attribut ou valeur d'élément

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser les opérateurs parenthèses— `(` et `)` —pour filtrer des éléments avec un nom d'élément spécifique ou une valeur d'attribut. Considérez l'objet XML suivant :

```
var x:XML =
    <employeeList>
        <employee id="347">
            <lastName>Zmed</lastName>
            <firstName>Sue</firstName>
            <position>Data analyst</position>
        </employee>
        <employee id="348">
            <lastName>McGee</lastName>
            <firstName>Chuck</firstName>
            <position>Jr. data analyst</position>
        </employee>
    </employeeList>
```

Les expressions suivantes sont toutes valides :

- `x.employee.(lastName == "McGee")`—Il s'agit du deuxième nœud `employee`.
- `x.employee.(lastName == "McGee").firstName`—Il s'agit de la propriété `firstName` du deuxième nœud `employee`.
- `x.employee.(lastName == "McGee").@id`—Il s'agit de la valeur de l'attribut `id` du deuxième nœud `employee`.
- `x.employee.(@id == 347)`—Le premier nœud `employee`.
- `x.employee.(@id == 347).lastName`—Il s'agit de la propriété `lastName` du premier nœud `employee`.
- `x.employee.(@id > 300)`—Il s'agit d'un XMLList avec deux propriétés `employee`.
- `x.employee.(position.toString().search("analyst") > -1)`—Il s'agit d'un XMLList avec deux propriétés `position`.

Si vous tentez d'appliquer un filtre à des attributs ou des éléments qui n'existent pas, une exception est renvoyée. Par exemple, la dernière ligne du code suivant génère une erreur car il n'existe aucun attribut `id` dans le deuxième élément `p` :

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

De même, la dernière ligne du code suivant génère une erreur car il n'existe aucune propriété `b` du deuxième élément `p` :

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

Pour éviter ces erreurs, vous pouvez identifier les propriétés ayant les éléments ou les attributs correspondants, à l'aide des méthodes `attribute()` et `elements()`, comme dans le code suivant :

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(attribute('id') == '123'));
trace(doc.p.(elements('b') == 'Bob'));
```

Vous pouvez également utiliser la méthode `hasOwnProperty()`, comme dans le code suivant :

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(hasOwnProperty('@id') && @id == '123'));
trace(doc.p.(hasOwnProperty('b') && b == 'Bob'));
```

Utilisation de for..in et for each.. dans les instructions

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 propose les instructions `for..in` et `for each..in` pour permettre les itérations dans les objets XMLList. Par exemple, considérez l'objet XML suivant, `myXML`, et l'objet XMLList, `myXML.item`. L'objet XMLList, `myXML.item`, est constitué de deux nœuds `item` de l'objet XML.

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
        <item id='2' quantity='2'>
            <menuName>fries</menuName>
            <price>1.45</price>
        </item>
    </order>;
```

La boucle `for..in` permet de procéder à une itération sur un ensemble de noms de propriété dans un objet XMLList :

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

La boucle `for each..in` permet de procéder à une itération dans les propriétés de l'objet XMLList :

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

Utilisation des espaces de noms XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les espaces de noms dans un objet (ou document) XML identifient le type de données que contient l'objet. Par exemple, lorsque vous envoyez et fournissez des données XML à un service Web qui utilise le protocole de messagerie SOAP, vous déclarez l'espace de noms dans la balise de début de l'XML :

```
var message:XML =
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <soap:Body xmlns:w="http://www.test.com/weather/">
            <w:getWeatherResponse>
                <w:tempurature >78</w:tempurature>
            </w:getWeatherResponse>
        </soap:Body>
    </soap:Envelope>;
```

L'espace de nom a un préfixe, `soap`, et un URI qui le définit, `http://schemas.xmlsoap.org/soap/envelope/`.

ActionScript 3.0 inclut la classe Namespace pour utiliser des espaces de noms XML. Pour l'objet XML de l'exemple précédent, vous pouvez utiliser la classe Namespace comme suit :

```
var soapNS:Namespace = message.namespace("soap");
trace(soapNS); // Output: http://schemas.xmlsoap.org/soap/envelope/

var wNS:Namespace = new Namespace("w", "http://www.test.com/weather/");
message.addNamespace(wNS);
var encodingStyle:XMLList = message.@soapNS::encodingStyle;
var body:XMLList = message.soapNS::Body;

message.soapNS::Body.wNS::GetWeatherResponse.wNS::tempurature = "78";
```

Les méthodes suivantes de la classe XML permettent d'utiliser les espaces de nom : `addNamespace()`, `inScopeNamespaces()`, `localName()`, `name()`, `namespace()`, `namespaceDeclarations()`, `removeNamespace()`, `setLocalName()`, `setName()` et `setNamespace()`.

La directive `default xml namespace` vous permet d'affecter un espace de nom par défaut associé aux objets XML. Par exemple, dans l'exemple suivant, `x1` et `x2` partagent le même espace de nom par défaut :

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

Conversion de type XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez convertir des objets XML et XMLList en valeurs String. De même, vous pouvez convertir des chaînes en objets XML et XMLList. Sachez que toutes les valeurs d'attribut, les noms et les valeurs de texte XML sont des chaînes. Les sections suivantes décrivent toutes ces formes de conversion de type XML.

Conversion d'objets XML et XMLList en chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les classes XML et XMLList contiennent les méthodes `toString()` et `toXMLString()`. La méthode `toXMLString()` renvoie une chaîne qui comprend la totalité des balises, des attributs, des déclarations d'espace de nom et du contenu de l'objet XML. Pour les objets XML ayant un contenu complexe (éléments enfant), la méthode `toString()` procède exactement comme la méthode `toXMLString()`. Pour les objets XML ayant un contenu simple (ceux qui contiennent un seul élément de texte), la méthode `toString()` renvoie uniquement le contenu de texte de l'élément, comme indiqué dans l'exemple suivant :

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName.toXMLString());
    // <menuName>burger</menuName>
trace(myXML.item[0].menuName.toString());
    // burger
```

Si vous utilisez la méthode `trace()` sans spécifier `toString()` ou `toXMLString()`, les données sont converties à l'aide de la méthode `toString()` par défaut, comme indiqué dans le code suivant :

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

Lorsque vous utilisez la méthode `trace()` pour déboguer un code, vous pouvez utiliser la méthode `toXMLString()` de façon à ce que la méthode `trace()` génère des données plus complètes.

Conversion de chaînes en objets XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser le constructeur `new XML()` pour créer un objet XML à partir d'une chaîne, comme suit :

```
var x:XML = new XML("<a>test</a>");
```

Si vous tentez de convertir une chaîne en XML à partir d'une chaîne qui représente un XML non valide ou un XML qui n'est pas formé correctement, une erreur d'exécution est renvoyée, comme suit :

```
var x:XML = new XML("<a>test"); // throws an error
```

Conversion de valeurs d'attribut, de noms et de valeurs de texte à partir de chaînes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Toutes les valeurs d'attribut XML, les noms et les valeurs de texte sont des types de données String que vous pouvez convertir en d'autres types de données. Par exemple, le code suivant utilise la fonction `Number()` pour convertir des valeurs de texte en nombres :

```
var myXML:XML =
                        <order>
                            <item>
                                <price>3.95</price>
                            </item>
                            <item>
                                <price>1.00</price>
                            </item>
                        </order>;

var total:XML = <total>0</total>;
myXML.appendChild(total);

for each (var item:XML in myXML.item)
{
    myXML.total.children()[0] = Number(myXML.total.children()[0])
                                                + Number(item.price.children()[0]);
}
trace(myXML.total); // 4.95;
```

Si ce code n'utilisait pas la fonction `Number()`, il interpréterait l'opérateur + comme l'opérateur de concaténation de chaîne, et la méthode `trace()` de la dernière ligne générerait les éléments suivants :

```
01.003.95
```

Lecture de documents XML externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la classe URLLoader pour charger des données XML depuis une URL. Pour utiliser le code suivant dans vos applications, remplacez la valeur `XML_URL` dans l'exemple par une URL valide :

```
import flash.events.Event;
import flash.net.URLLoader;

var myXML:XML = new XML();
var XML_URL:String = "http://www.example.com/Sample3.xml";
var myXMLURL:URLRequest = new URLRequest(XML_URL);
var myLoader:URLLoader = new URLLoader(myXMLURL);
myLoader.addEventListener(Event.COMPLETE, xmlLoaded);

function xmlLoaded(event:Event):void
{
    myXML = XML(myLoader.data);
    trace("Data loaded.");
}
```

Vous pouvez également utiliser la classe XMLSocket pour définir une connexion de socket XML asynchrone avec un serveur. Pour plus d'informations, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Utilisation de XML dans un exemple ActionScript : chargement de données RSS depuis Internet

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple d'application RSSViewer présente plusieurs fonctions d'utilisation d'XML dans ActionScript, notamment :

- Utilisation de méthodes XML pour parcourir des données XML sous la forme d'un flux RSS.
- Utilisation de méthodes XML pour assembler des données XML sous la forme HTML à utiliser dans un champ de texte.

Le format RSS est largement utilisé pour diffuser des nouvelles via XML. Un fichier de données RSS simple peut avoir l'aspect suivant :

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
    <title>Alaska - Weather</title>
    <link>http://www.nws.noaa.gov/alerts/ak.html</link>
    <description>Alaska - Watches, Warnings and Advisories</description>

    <item>
        <title>
            Short Term Forecast - Taiya Inlet, Klondike Highway (Alaska)
        </title>
        <link>
            http://www.nws.noaa.gov/alerts/ak.html#A18.AJKNK.1900
        </link>
        <description>
            Short Term Forecast Issued At: 2005-04-11T19:00:00
            Expired At: 2005-04-12T01:00:00 Issuing Weather Forecast Office
            Homepage: http://pajk.arh.noaa.gov
        </description>
</item>
    <item>
        <title>
            Short Term Forecast - Haines Borough (Alaska)
        </title>
            <link>
            http://www.nws.noaa.gov/alerts/ak.html#AKZ019.AJKNOWAJK.190000
        </link>
        <description>
            Short Term Forecast Issued At: 2005-04-11T19:00:00
            Expired At: 2005-04-12T01:00:00 Issuing Weather Forecast Office
            Homepage: http://pajk.arh.noaa.gov
        </description>
    </item>
</channel>
</rss>
```

L'application SimpleRSS lit les données RSS depuis Internet, analyse les données à la recherche de titres, de liens et de descriptions et renvoie ces données. La classe SimpleRSSUI fournit l'IU et appelle la classe SimpleRSS qui effectue le traitement XML.

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application RSSViewer se trouvent dans le dossier Samples/RSSViewer. L'application se compose des fichiers suivants :

Fichier	Description
RSSViewer.mxml ou RSSViewer.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/rssViewer/RSSParser.as	Une classe qui contient des méthodes utilisant E4X pour parcourir des données (XML) RSS et générer une représentation HTML correspondante.
RSSData/ak.rss	Un exemple de fichier RSS. Cette application est définie pour lire des données RSS à partir du Web, sur un flux RSS Flex hébergé par Adobe. Néanmoins, vous pouvez facilement modifier l'application pour lire des données RSS depuis ce document qui utilise un schéma légèrement différent de celui du flux RSS Flex.

Lecture et analyse de données XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe RSSParser comprend une méthode `xmlLoaded()` qui convertit les données RSS d'entrée, stockées dans la variable `rssXML`, en une chaîne contenant une sortie formatée HTML, `rssOutput`.

Vers le début de la méthode, le code définit l'espace de nom XML par défaut si les données RSS source comprennent un espace de nom par défaut :

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

Les lignes suivantes parcourent ensuite en boucle le contenu des données XML source, en examinant chaque propriété descendante appelée `item` :

```
for each (var item:XML in rssXML..item)
{
    var itemTitle:String = item.title.toString();
    var itemDescription:String = item.description.toString();
    var itemLink:String = item.link.toString();
    outXML += buildItemHTML(itemTitle,
                            itemDescription,
                            itemLink);
}
```

Les trois premières lignes définissent simplement des variables de chaîne pour représenter les propriétés de titre, de description et de lien de la propriété `item` des données XML. La ligne suivante appelle la méthode `buildItemHTML()` pour obtenir des données HTML sous la forme d'un objet XMLList et utilise les trois nouvelles variables de chaîne en tant que paramètres.

Assemblage de données XMLList

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les données HTML (un objet XMLList) se présentent comme suit :

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

Les premières lignes de la méthode effacent l'espace de noms xml par défaut :

```
default xml namespace = new Namespace();
```

La directive `default xml namespace` a un domaine de niveau bloc de fonction. Cela signifie que le domaine de cette déclaration est la méthode `buildItemHTML()`.

Les lignes qui suivent assemblent l'objet XMLList en fonction des arguments de chaîne transmis à la fonction :

```
var body:XMLList = new XMLList();
body += new XML("<b>" + itemTitle + "</b>");
var p:XML = new XML("<p>" + itemDescription + "</p>");

var link:XML = <a></a>;
link.@href = itemLink; // <link href="itemLinkString"></link>
link.font.@color = "#008000";
        // <font color="#008000"></font></a>
        // 0x008000 = green
link.font = "More...";

p.appendChild(<br/>);
p.appendChild(link);
body += p;
```

Cet objet XMLList représente des données de chaîne adaptées à un champ de texte HTML d'ActionScript.

La méthode `xmlLoaded()` utilise la valeur de renvoi de la méthode `buildItemHTML()` et la convertit en une chaîne :

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

Extraction du titre du flux RSS et envoi d'un événement personnalisé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `xmlLoaded()` définit une variable de chaîne `rssTitle` en fonction des informations contenues dans les données XML RSS source :

```
rssTitle = rssXML.channel.title.toString();
```

Pour finir, la méthode `xmlLoaded()` génère un événement qui notifie l'application que les données sont analysées et disponibles :

```
            dataWritten = new Event("dataWritten", true);
```

Chapitre 7 : Utilisation de la fonctionnalité JSON native

ActionScript 3.0 fournit une API native permettant de coder et de décoder des objets ActionScript à l'aide du format JavaScript Object Notation (JSON). La classe JSON et les fonctions membres associées suivent la spécification ECMA-262 5e édition avec de légères variations.

Le membre de la communauté Todd Anderson fournit une comparaison entre l'API JSON native et la classe JSON as3corelib tierce. Voir Working with Native JSON in Flash Player 11 (disponible en anglais uniquement).

Voir aussi

JSON

Présentation de l'API JSON

L'API JSON ActionScript est constituée de la classe JSON et des fonctions membres `toJSON()` sur quelques classes natives. Pour les applications nécessitant un codage JSON personnalisé pour une classe, la structure ActionScript propose diverses méthodes permettant de remplacer le codage par défaut.

La classe JSON gère en interne l'importation et l'exportation des classes ActionScript qui ne fournissent pas de membre `toJSON()`. Dans ces cas, la classe JSON traverse les propriétés publiques de chaque objet qu'elle détecte. Si un objet contient d'autres objets, JSON parcourt les objets imbriqués et effectue la même traversée. Si un objet fournit une méthode `toJSON()`, JSON utilise cette méthode personnalisée plutôt que son algorithme interne.

L'interface JSON est composée d'une méthode d'encodage, `stringify()`, et d'une méthode de décodage, `parse()`. Chacune de ces méthodes fournit un paramètre permettant d'insérer votre propre logique dans la procédure de codage et de décodage JSON. Pour `stringify()`, ce paramètre est appelé `replacer` ; pour `parse()`, il est appelé `reviver`. Ces paramètres prennent une définition de fonction avec deux arguments à l'aide de la signature suivante :

```
function(k, v):*
```

Méthodes toJSON()

La signature des méthodes `toJSON()` est

```
public function toJSON(k:String):*
```

`JSON.stringify()` appelle la méthode `toJSON()`, si elle existe, pour chaque propriété publique qu'elle rencontre lorsqu'elle traverse un objet. Une propriété consiste en une paire clé-valeur. Lorsque `stringify()` appelle la méthode `toJSON()`, il transmet la clé, `k`, de la propriété qu'il examine actuellement. Une implémentation `toJSON()` standard évalue chaque nom de propriété et renvoie l'encodage souhaité de sa valeur.

La méthode `toJSON()` peut renvoyer tout type de valeur (identifiée à l'aide du signe *) et pas uniquement une chaîne. Ce type de renvoi de variable permet à la méthode `toJSON()` de renvoyer un objet, le cas échéant. Par exemple, si une propriété de votre classe personnalisée contient un objet issu d'une autre bibliothèque tierce, vous pouvez renvoyer cet objet dès que la méthode `toJSON()` détecte votre propriété. JSON parcourt alors l'objet tiers. Le processus de codage est le suivant :

- Si la méthode `toJSON()` renvoie un objet qui n'est pas évalué par rapport à une chaîne, `stringify()` parcourt cet objet.
- Si la méthode `toJSON()` renvoie une chaîne, `stringify()` enveloppe la valeur dans une autre chaîne, renvoie la chaîne enveloppée et passe à la valeur suivante.

Dans de nombreux cas, il est préférable de renvoyer un objet plutôt que de renvoyer une chaîne JSON créée par votre application. Le renvoi d'un objet implique l'utilisation de l'algorithme de codage JSON et permet à JSON de se répéter dans les objets imbriqués.

La méthode `toJSON()` n'est pas définie dans la classe Object ou dans la plupart des autres classes natives. Son absence indique à JSON d'effectuer sa traversée standard sur les propriétés publiques de l'objet. Si vous préférez, vus pouvez également utiliser la méthode `toJSON()` pour exposer les propriétés privées de votre objet.

Certaines classes natives posent néanmoins des problèmes que les bibliothèques ActionScript sont incapables de résoudre efficacement dans tous les cas d'utilisation. Pour ces classes, ActionScript fournit une implémentation triviale que le client peut à nouveau implémenter selon ses besoins. Les classes qui fournissent des membres `toJSON()` triviaux sont les suivantes :

- ByteArray
- Date
- Dictionary
- XML

Vous pouvez intégrer la classe ByteArray dans une sous-classe pour remplacer sa méthode `toJSON()`, mais vous pouvez aussi redéfinir son prototype. Les classes Date et XML, qui sont déclarées comme étant finales, vous obligent à utiliser le prototype de classe pour redéfinir `toJSON()`. La classe Dictionary est déclarée comme étant dynamique, ce qui vous donne la liberté de remplacer la méthode `toJSON()`.

Définition du comportement JSON personnalisé

Vous disposez de plusieurs options pour implémenter vos propres codage et décodage JSON pour les classes natives :

- Définir ou remplacer `toJSON()` sur la sous-classe personnalisée d'une classe native non finale
- Définir et redéfinir `toJSON()` sur le prototype de la classe
- Définir une propriété `toJSON` sur une classe dynamique
- Utiliser les paramètres `JSON.stringify() replacer` et `JSON.parser() reviver`

Voir aussi

ECMA-262, 5e édition

Définition de la méthode toJSON() sur le prototype d'une classe intégrée

L'implémentation JSON native dans ActionScript imite le mécanisme JSON ECMAScript défini dans ECMA-262, 5e édition. Etant donné qu'ECMAScript ne prend pas en charge les classes, ActionScript définit le comportement de JSON en termes de distribution basée sur les prototypes. Ancêtres des classes ActionScript 3.0, les prototypes permettent un héritage simulé, ainsi que les ajouts et redéfinitions de membres.

ActionScript permet de définir ou de redéfinir `toJSON()` sur le prototype d'une classe. Ce privilège s'étend aux classes déclarées comme étant finales. Lorsque vous définissez `toJSON()` sur le prototype d'une classe, votre définition s'applique à toutes les occurrences de cette classe dans le cadre de votre application. Par exemple, voici comment vous pouvez définir une méthode `toJSON()` sur le prototype de la classe MovieClip :

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

Lorsque votre application appelle la méthode `stringify()` sur une occurrence de MovieClip, `stringify()` renvoie le résultat de votre méthode `toJSON()` :

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

Vous pouvez en outre remplacer `toJSON()` dans les classes natives qui définissent cette méthode. Par exemple, le code suivant remplace `Date.toJSON()` :

```
Date.prototype.toJSON = function (k):* {
    return "any date format you like via toJSON: "+
        "this.time:"+this.time + " this.hours:"+this.hours;
}
var dt:Date = new Date();
trace(JSON.stringify(dt));
// "any date format you like via toJSON: this.time:1317244361947 this.hours:14"
```

Définition ou remplacement de toJSON() au niveau de la classe

Les applications n'ont pas toujours besoin d'utiliser des prototypes pour redéfinir `toJSON()`. Il est également possible de définir `toJSON()` en tant que membre d'une sous-classe si la classe parente n'est pas marquée comme finale. Vous pouvez par exemple étendre la classe ByteArray et définir une fonction `toJSON()` publique :

```
package {

    import flash.utils.ByteArray;
    public class MyByteArray extends ByteArray
    {
        public function MyByteArray() {
        }

        public function toJSON(s:String):*
        {
            return "MyByteArray";
        }

    }
}


var ba:ByteArray = new ByteArray();
trace(JSON.stringify(ba)); //"ByteArray"
var mba:MyByteArray = new MyByteArray(); //"MyByteArray"
trace(JSON.stringify(mba)); //"MyByteArray"
```

Si un classe est dynamique, il est possible d'ajouter une propriété `toJSON` à un objet de cette classe et de lui attribuer une fonction de la façon suivante :

```
var d:Dictionary = new Dictionary();
trace(JSON.stringify((d))); // "Dictionary"
d.toJSON = function(){return {c : "toJSON override."};} // overrides existing function
trace(JSON.stringify((d))); // {"c":"toJSON override."}
```

Vous pouvez remplacer, définir ou redéfinir `toJSON()` sur n'importe quelle classe ActionScript. Néanmoins, la plupart des classes ActionScript intégrées ne définissent pas `toJSON()`. La classe Object ne définit pas la méthode `toJSON` dans son prototype par défaut ni ne la déclare en tant que membre de classe. Seule une poignée de classes natives définit la méthode comme fonction prototype. C'est pourquoi, dans la plupart des cas, vous ne pouvez pas remplacer `toJSON()` de façon traditionnelle.

Les classes natives qui ne définissent pas `toJSON()` sont sérialisées sur JSON par l'implémentation JSON interne. Dans la mesure du possible, évitez de remplacer cette fonctionnalité intégrée. Si vous définissez un membre `toJSON()`, la classe JSON utilise votre logique plutôt que sa propre fonctionnalité.

Utilisation du paramètre replacer de la méthode JSON.stringify()

Il peut être utile de remplacer `toJSON()` sur le prototype en vue de modifier le comportement d'exportation JSON d'une classe dans une application. Néanmoins, votre logique d'exportation doit s'appliquer uniquement à des cas spéciaux sous des conditions provisoires. Pour prendre en compte ces modifications à petite échelle, vous pouvez utiliser le paramètre `replacer` de la méthode `JSON.stringify()`.

La méthode `stringify()` applique la fonction transmise via le paramètre `replacer` à l'objet en cours de codage. La signature pour cette fonction est similaire à celle de `toJSON()` :

```
function (k,v):*
```

Contrairement à la méthode `toJSON()`, la fonction `replacer` requiert la valeur `v`, ainsi que la clé `k`. Cette différence est nécessaire, car la méthode `stringify()` est définie sur l'objet JSON statique et non sur l'objet en cours de codage. Lorsque la méthode `JSON.stringify()` appelle `replacer(k,v)`, elle traverse l'objet d'entrée d'origine. Le paramètre implicite `this` transmis à la fonction `replacer` fait référence à l'objet qui détient la clé et la valeur. Etant donné que la méthode `JSON.stringify()` ne modifie pas l'objet d'entrée d'origine, cet objet reste inchangé dans le conteneur actuellement traversé. Vous pouvez par conséquent utiliser le code `this[k]` pour interroger la clé sur l'objet d'origine. Le paramètre `v` renferme la valeur que `toJSON()` convertit.

Tout comme `toJSON()`, la fonction `replacer` peut renvoyer tout type de valeur. Si `replacer` renvoie une chaîne, le moteur JSON convertit le contenu en séquence d'échappement entre guillemets et place ce contenu également entre guillemets. Cette structure garantit que `stringify()` reçoive un objet de chaîne JSON valide qui reste une chaîne dans un prochain appel de `JSON.parse()`.

Le code suivant utilise le paramètre `replacer` et le paramètre `this` implicite pour renvoyer les valeurs `time` et `hours` d'un objet Date :

```
JSON.stringify(d, function (k,v):* {
    return "any date format you like via replacer: "+
        "holder[k].time:"+this[k].time + " holder[k].hours:"+this[k].hours;
});
```

Utilisation du paramètre reviver de la méthode JSON.parse()

Le paramètre `reviver` de la méthode `JSON.parse()` est l'opposé de la fonction `replacer` : il convertit une chaîne JSON en objet ActionScript utilisable. L'argument `reviver` est une fonction qui prend deux paramètres et renvoie tout type :

```
function (k,v):*
```

Dans cette fonction, `k` est une clé et `v` est la valeur de `k`. Tout comme `stringify()`, `parse()` traverse les paires clé-valeur JSON et applique la fonction`reviver`, si elle existe, à chaque paire. L'un des problèmes potentiels est que la classe JSON ne renvoie pas le nom de classe ActionScript d'un objet. Il peut par conséquent être difficile de savoir quel type d'objet ranimer. Ce problème peut en outre s'avérer particulièrement délicat lorsque les objets sont imbriqués. En désignant les fonctions `toJSON()`, `replacer` et `reviver`, vous pouvez trouver des moyens d'identifier les objets ActionScript exportés tout en gardant intacts les objets d'origine.

Exemple d'analyse

L'exemple suivant illustre une stratégie de ranimation d'objets analysés à partir de chaînes JSON. Cet exemple définit deux classes : JSONGenericDictExample et JSONDictionaryExtnExample. La classe JSONGenericDictExample est une classe Dictionary personnalisée. Chaque enregistrement contient le nom et la date de naissance d'une personne, ainsi qu'un ID unique. Chaque fois que le constructeur JSONGenericDictExample est appelé, il ajoute l'objet nouvellement créé à un tableau statique interne avec un entier augmentant statiquement comme son ID. La classe JSONGenericDictExample définit également une méthode `revive()` qui extrait uniquement l'entier du membre `id` le plus long. La méthode `revive()` utilise cet entier pour rechercher et renvoyer l'objet ranimable adéquat.

La classe JSONDictionaryExtnExample étend la classe Dictionary ActionScript. Ses enregistrements n'ont pas de structure définie et peuvent contenir toutes sortes de données. Les données sont attribuées après la construction de l'objet JSONDictionaryExtnExample et non par les propriétés définies dans la classe. Les enregistrements de JSONDictionaryExtnExample utilisent les objets JSONGenericDictExample comme clés. Lorsqu'un objet JSONDictionaryExtnExample est ranimé, la fonction `JSONGenericDictExample.revive()` utilise l'ID associé à JSONDictionaryExtnExample pour récupérer l'objet de clé correct.

Plus important encore, la méthode `JSONDictionaryExtnExample.toJSON()` renvoie une chaîne de marqueurs en plus de l'objet JSONDictionaryExtnExample. Cette chaîne identifie la sortie JSON comme appartenant à la classe JSONDictionaryExtnExample. Ce marqueur indique clairement le type d'objet en cours de traitement lors de l'exécution de `JSON.parse()`.

```
package {
    // Generic dictionary example:
    public class JSONGenericDictExample {
        static var revivableObjects = [];
        static var nextId = 10000;
        public var id;
        public var dname:String;
        public var birthday;

        public function JSONGenericDictExample(name, birthday) {
            revivableObjects[nextId] = this;
            this.id       = "id_class_JSONGenericDictExample_" + nextId;
            this.dname     = name;
            this.birthday = birthday;
            nextId++;
        }
        public function toString():String { return this.dname; }
        public static function revive(id:String):JSONGenericDictExample {
            var r:RegExp = /^id_class_JSONGenericDictExample_([0-9]*)$/;
            var res = r.exec(id);
            return JSONGenericDictExample.revivableObjects[res[1]];
        }
    }
}

package {
    import flash.utils.Dictionary;
    import flash.utils.ByteArray;

    // For this extension of dictionary, we serialize the contents of the
    // dictionary by using toJSON
    public final class JSONDictionaryExtnExample extends Dictionary {
        public function toJSON(k):* {
            var contents = {};
            for (var a in this) {
                contents[a.id] = this[a];
            }

            // We also wrap the contents in an object so that we can
            // identify it by looking for the marking property "class E"
            // while in the midst of JSON.parse.
```

```
            return {"class JSONDictionaryExtnExample": contents};
        }

        // This is just here for debugging and for illustration
        public function toString():String {
            var retval = "[JSONDictionaryExtnExample <";
            var printed_any = false;
            for (var k in this) {
                retval += k.toString() + "=" +
                "[e="+this[k].earnings +
                ",v="+this[k].violations + "], "
                printed_any = true;
            }
            if (printed_any)
                retval = retval.substring(0, retval.length-2);
            retval += ">]"
            return retval;
        }
    }
}
```

Lorsque le script d'exécution suivant appelle `JSON.parse()` sur un objet JSONDictionaryExtnExample, la fonction `reviver` appelle `JSONGenericDictExample.revive()` sur chaque objet dans JSONDictionaryExtnExample. Cet appel extrait l'ID qui représente la clé de l'objet. La fonction `JSONGenericDictExample.revive()` utilise cet ID pour récupérer et renvoyer l'objet JSONDictionaryExtnExample stocké à partir d'un tableau statique privé.

```
import flash.display.MovieClip;
import flash.text.TextField;

var a_bob1:JSONGenericDictExample = new JSONGenericDictExample("Bob", new
Date(Date.parse("01/02/1934")));
var a_bob2:JSONGenericDictExample = new JSONGenericDictExample("Bob", new
Date(Date.parse("05/06/1978")));
var a_jen:JSONGenericDictExample = new JSONGenericDictExample("Jen", new
Date(Date.parse("09/09/1999")));

var e = new JSONDictionaryExtnExample();
e[a_bob1] = {earnings: 40, violations: 2};
e[a_bob2] = {earnings: 10, violations: 1};
e[a_jen]  = {earnings: 25, violations: 3};

trace("JSON.stringify(e): " + JSON.stringify(e)); // {"class JSONDictionaryExtnExample":
                        //{"id_class_JSONGenericDictExample_10001":
                        //{"earnings":10,"violations":1},
                        //"id_class_JSONGenericDictExample_10002":
                        //{"earnings":25,"violations":3},
                        //"id_class_JSONGenericDictExample_10000":
                        // {"earnings":40,"violations":2}}}

var e_result = JSON.stringify(e);

var e1 = new JSONDictionaryExtnExample();
var e2 = new JSONDictionaryExtnExample();

// It's somewhat easy to convert the string from JSON.stringify(e) back
// into a dictionary (turn it into an object via JSON.parse, then loop
// over that object's properties to construct a fresh dictionary).
```

```
//
// The harder exercise is to handle situations where the dictionaries
// themselves are nested in the object passed to JSON.stringify and
// thus does not occur at the topmost level of the resulting string.
//
// (For example: consider roundtripping something like
//   var tricky_array = [e1, [[4, e2, 6]], {table:e3}]
// where e1, e2, e3 are all dictionaries.  Furthermore, consider
// dictionaries that contain references to dictionaries.)
//
// This parsing (or at least some instances of it) can be done via
// JSON.parse, but it's not necessarily trivial.  Careful consideration
// of how toJSON, replacer, and reviver can work together is
// necessary.

var e_roundtrip =
    JSON.parse(e_result,
            // This is a reviver that is focused on rebuilding JSONDictionaryExtnExample objects.
                function (k, v) {
                     if ("class JSONDictionaryExtnExample" in v) { // special marker tag;
                         //see JSONDictionaryExtnExample.toJSON().
                        var e = new JSONDictionaryExtnExample();
                        var contents = v["class JSONDictionaryExtnExample"];
                        for (var i in contents) {
                           // Reviving JSONGenericDictExample objects from string
                           // identifiers is also special;
                           // see JSONGenericDictExample constructor and
                           // JSONGenericDictExample's revive() method.
                            e[JSONGenericDictExample.revive(i)] = contents[i];
                        }
                        return e;
                    } else {
                        return v;
                    }
                });

trace("// == Here is an extended Dictionary that has been round-tripped  ==");
trace("// == Note that we have revived Jen/Jan during the roundtrip.    ==");
trace("e:           " + e); //[JSONDictionaryExtnExample <Bob=[e=40,v=2], Bob=[e=10,v=1],
                            //Jen=[e=25,v=3]>]
trace("e_roundtrip: " + e_roundtrip); //[JSONDictionaryExtnExample <Bob=[e=40,v=2],
                                      //Bob=[e=10,v=1], Jen=[e=25,v=3]>]
trace("Is e_roundtrip a JSONDictionaryExtnExample? " + (e_roundtrip is
JSONDictionaryExtnExample)); //true
trace("Name change: Jen is now Jan");
a_jen.dname = "Jan"

trace("e:           " + e); //[JSONDictionaryExtnExample <Bob=[e=40,v=2], Bob=[e=10,v=1],
                            //Jan=[e=25,v=3]>]
trace("e_roundtrip: " + e_roundtrip); //[JSONDictionaryExtnExample <Bob=[e=40,v=2],
                                      //Bob=[e=10,v=1], Jan=[e=25,v=3]>]
```

Chapitre 8 : Gestion des événements

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un système de gestion des événements permet au programmeur de répondre aux actions de l'utilisateur et aux événements système de manière pratique. Le modèle d'événement ActionScript 3.0 n'est pas seulement pratique, il est conforme aux normes en vigueur et bien intégré à la liste d'affichage. Ce modèle repose sur la spécification d'événements Document Object Model (DOM) de niveau 3, une architecture de gestion d'événements normalisée. Il constitue donc pour les programmeurs ActionScript un outil puissant et parfaitement intuitif.

Le système de gestion d'événements ActionScript 3.0 assure une interaction étroite avec la liste d'affichage. Pour comprendre les principes de base de la liste d'affichage, voir « Programmation de l'affichage » à la page 156.

Voir aussi

Package flash.events

Spécification d'événements Document Object Model (DOM) niveau 3

Principes de base de la gestion des événements

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez concevoir un événement comme tout type d'action qui se produit dans votre fichier SWF et qui présente un intérêt pour vous en tant que programmeur. Par exemple, la plupart des fichiers SWF prennent en charge une certaine forme d'interaction, qu'il s'agisse d'une action aussi simple qu'un clic avec la souris ou d'une opération plus complexe, telle que l'acceptation et le traitement des données saisies dans un formulaire. Toute interaction de ce type dans le fichier SWF est considérée comme un événement. Des événements peuvent également se produire sans aucun interaction directe de l'utilisateur, par exemple lorsque le chargement des données depuis un serveur se termine ou qu'une caméra reliée devient active.

Dans ActionScript 3.0, tout événement est représenté par un objet événement, qui correspond à une occurrence de la classe Event ou de l'une de ses sous-classes. Le rôle d'un objet événement est non seulement de stocker des informations relatives à un événement spécifique, mais aussi de contenir des méthodes qui favorisent la manipulation de cet objet. Par exemple, lorsque Flash Player ou AIR détecte un clic de la souris, il crée un objet événement (une occurrence de la classe MouseEvent) qui représente cet événement particulier.

Après la création d'un objet événement, Flash Player ou AIR le *distribue*, ce qui signifie que l'objet événement est transmis à l'objet représentant la cible de l'événement. L'objet qui doit recevoir l'objet événement ainsi distribué est appelé *cible d'événement*. Par exemple, lorsqu'une caméra reliée devient active, Flash Player distribue un objet événement directement à la cible de l'événement, dans ce cas l'objet représentant la caméra. Toutefois, si la cible d'événement se trouve dans la liste d'affichage, l'objet événement est transmis tout au long de la hiérarchie de la liste d'affichage jusqu'à ce qu'il atteigne la cible en question. Dans certains cas, l'objet événement se « propage » ensuite vers le haut de la hiérarchie de la liste d'affichage, selon le même cheminement. Cette traversée de la hiérarchie de la liste d'affichage correspond au *flux d'événements*.

Vous pouvez « écouter » les objets événement de votre code grâce aux écouteurs d'événement. Les *écouteurs d'événement* sont des fonctions ou des méthodes que vous écrivez pour répondre aux différents événements. Pour garantir que le programme réagisse aux événements, vous devez ajouter des écouteurs d'événement soit à la cible d'événement, soit à l'un des objets de la liste d'affichage qui font partie du flux d'événements de l'objet événement.

Chaque fois que vous écrivez un code d'écouteur d'événement, il suit cette structure de base (les éléments en gras sont des espaces réservés que vous rempliriez pour votre cas particulier) :

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

Ce code a un double rôle. Tout d'abord, il définit une fonction, qui est une manière de spécifier les actions à exécuter en réponse à l'événement. Ensuite, il appelle la méthode `addEventListener()` de l'objet source, « inscrivant » ainsi la fonction auprès de l'événement spécifié de sorte que lorsque l'événement survient, les actions de la fonction ont lieu. Lorsque l'événement se produit, la cible de l'événement vérifie sa liste de toutes les fonctions et méthodes enregistrées en tant qu'écouteurs d'événement. Elle appelle ensuite chacune d'elles, transmettant l'objet événement à titre de paramètre.

Vous devez apporter quatre modifications à ce code pour créer votre propre écouteur d'événement. Premièrement, vous devez remplacer le nom de la fonction par celui que vous souhaitez utiliser (ceci doit être modifié à deux endroits, là où le code indique **eventResponse**). Deuxièmement, vous devez spécifier le nom de la classe de l'objet événement qui est envoyé par l'événement que vous souhaitez écouter (**EventType** dans le code), et vous devez indiquer la constante pour l'événement en question (**EVENT_NAME** dans la liste). Troisièmement, vous devez appeler la méthode `addEventListener()` sur l'objet qui enverra l'événement (**eventTarget** dans ce code). Vous pouvez également modifier le nom de la variable utilisée comme paramètre de la fonction (**eventObject** dans ce code).

Concepts importants et terminologie

La liste de référence suivante contient des termes importants utilisés dans le cadre de la rédaction de routines de gestion des événements :

Propagation Certains événements donnent lieu à une propagation afin de permettre à un objet d'affichage parent de réagir aux événements distribués par ses enfants.

Phase de propagation Partie du flux d'événements dans laquelle un événement est propagé jusqu'aux objets d'affichage parent. La phase de propagation suit la phase de capture et la phase cible.

Phase de capture Partie du flux d'événements dans laquelle un événement se propage de la cible la plus générale vers l'objet cible le plus spécifique. La phase de capture précède la phase cible et la phase de propagation.

Comportement par défaut Certains événements sont liés à un comportement appelé comportement par défaut. Par exemple, lorsqu'un utilisateur tape du texte dans un champ, un événement de saisie de texte est déclenché. Le comportement par défaut de cet événement consiste à afficher le caractère tapé dans le champ de texte—mais vous pouvez annuler ce comportement par défaut (si vous ne souhaitez pas afficher le caractère tapé, par exemple).

Distribuer Indiquer à des écouteurs d'événements qu'un événement a eu lieu.

Evénement Opération subie par un objet et que ce dernier peut signaler à d'autres objets.

Flux d'événements Lorsque des événements concernent un objet de la liste d'affichage (un objet affiché à l'écran), tous les objets qui contiennent cet objet sont informés de l'événement et avertissent à leur tour les écouteurs d'événements correspondants. Ce processus commence avec la scène et se poursuit à travers la liste d'affichage jusqu'à l'objet réel où s'est produit l'événement. Il recommence ensuite avec la scène. Ce processus est appelé flux d'événements.

Objet événement Objet contenant des informations sur l'occurrence d'un événement particulier, qui est envoyé à tous les écouteurs lorsqu'un événement est distribué.

Cible d'événement Objet qui envoie un événement. Par exemple, si l'utilisateur clique sur un bouton situé dans un Sprite se trouvant dans la scène, tous ces objets envoient des événements mais c'est au niveau de la cible d'événement que se produit l'événement (le bouton cliqué, dans ce cas).

Ecouteur Objet ou fonction qui s'est enregistré auprès d'un objet pour indiquer qu'il doit être averti lorsqu'un événement spécifique se produit.

Phase cible Stade du flux d'événements où un événement atteint la cible la plus spécifique possible. La phase cible se produit entre la phase de capture et la phase de propagation.

Variation de la gestion d'événements dans ActionScript 3.0 par rapport aux versions antérieures

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En ce qui concerne la gestion des événements, la différence la plus évidente entre ActionScript 3.0 et les versions antérieures est qu'ActionScript 3.0 comprend un seul système de gestion des événements alors que les anciennes versions d'ActionScript en comptent plusieurs. Cette section commence par une présentation générale du fonctionnement de la gestion des événements dans les versions précédentes, puis étudie les nouveautés qu'apporte ActionScript 3.0 dans ce domaine.

Gestion des événements dans les versions précédentes d'ActionScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Antérieurement à ActionScript 3.0, le langage ActionScript fournissait plusieurs méthodes de gestion des événements :

- Les gestionnaires d'événement `on()`, qui peuvent se placer directement sur des occurrences Button et MovieClip
- Les gestionnaires d'événement `onClipEvent()`, qui peuvent se placer directement sur des occurrences MovieClip
- Des propriétés de fonction de rappel, telles que `XML.onload` et `Camera.onActivity`
- Des écouteurs d'événement, que vous pouvez enregistrer à l'aide de la méthode `addListener()`
- La classe UIEventDispatcher, qui implémentait partiellement le modèle d'événements DOM

Chacun de ces mécanismes présente des avantages et des inconvénients. Les gestionnaires `on()` et `onClipEvent()` sont simples d'utilisation, mais compliquent la maintenance des projets car il peut s'avérer difficile de localiser le code placé directement sur les boutons ou les clips. Les fonctions de rappel sont également faciles à implémenter, mais imposent une limite d'une seule fonction de rappel par événement. L'implémentation des écouteurs d'événement est plus complexe : ils nécessitent non seulement la création d'un objet et d'une fonction d'écouteur, mais aussi l'enregistrement de l'écouteur auprès de l'objet qui génère l'événement. Bien qu'elle accroisse le temps système nécessaire, cette solution vous permet de créer plusieurs objets écouteur et de tous les enregistrer pour le même événement.

Dans ActionScript 2.0, le développement des composants engendrait un modèle d'événements encore différent. Ce nouveau modèle, caractérisé par la classe UIEventDispatcher, reposait sur un sous-ensemble de la spécification d'événements DOM. Ainsi, pour les développeurs accoutumés à la gestion des événements de composant, le passage au nouveau modèle d'événements d'ActionScript 3.0 se fera sans trop de difficultés.

Malheureusement, si l'on constate des recoupements entre les divers modèles d'événements, il existe aussi des différences. Par exemple, dans ActionScript 2.0, certaines propriétés, telles que `TextField.onChanged`, peuvent s'utiliser soit comme fonction de rappel, soit comme écouteur d'événement. Toutefois, la syntaxe qui permet d'enregistrer les objets écouteurs varie selon que vous utilisez l'une des six classes qui prennent en charge les écouteurs ou la classe UIEventDispatcher. Pour les classes Key, Mouse, MovieClipLoader, Selection, Stage et TextField, vous utilisez la méthode `addListener()`, mais pour la gestion des événements de composant, vous utilisez une méthode appelée `addEventListener()`.

La multiplicité des modèles de gestion d'événements a fait naître une autre complexité : l'étendue de la fonction de gestionnaire d'événement variait largement en fonction du mécanisme utilisé. En d'autres termes, la signification du mot-clé `this` n'était pas cohérente sur l'ensemble des systèmes de gestion d'événements.

Gestion d'événements dans ActionScript 3.0

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 utilise pour la première fois un modèle de gestion d'événements qui vient remplacer les nombreux mécanismes qui existaient dans les précédentes versions du langage. Le nouveau modèle d'événements repose sur la spécification d'événements de niveau 3 DOM (Document Object Model). Bien que le format de fichier SWF ne suive pas spécifiquement la norme DOM, il existe suffisamment de similitudes entre la liste d'affichage et la structure du DOM pour permettre l'implémentation de ce modèle d'événements. Un objet de la liste d'affichage est semblable à un noeud de la structure hiérarchique du DOM ; dans ce chapitre, les termes *objet de liste d'affichage* et *noeud* sont d'ailleurs utilisés de façon interchangeable.

L'implémentation du modèle d'événements DOM dans Flash Player et AIR comprend un concept appelé « comportements par défaut ». Un *comportement par défaut* est une action que Flash Player ou AIR effectue comme conséquence normale de certains événements.

Comportements par défaut

Les développeurs se chargent normalement d'écrire le code qui permet de répondre aux événements. Dans certains cas, cependant, un comportement est si couramment associé à un événement que Flash Player ou AIR l'exécute automatiquement, sauf si le développeur ajoute du code pour annuler son exécution. Comme Flash Player ou AIR se livre automatiquement à cette opération, on parle de comportements par défaut.

Par exemple, lorsqu'un utilisateur entre du texte dans un objet TextField, il est si courant de voir s'afficher la saisie dans l'objet TextField en question que ce comportement est prédéfini dans Flash Player ou AIR. Si vous ne souhaitez pas conserver ce comportement par défaut, vous pouvez l'annuler à l'aide du système de gestion des événements. Lorsqu'un utilisateur entre du texte dans un objet TextField, Flash Player ou AIR crée une occurrence de la classe TextEvent afin de représenter cette saisie. Pour éviter que Flash Player ou AIR n'affiche le texte dans l'objet TextField, vous devez accéder à cette occurrence de TextEvent spécifique et appeler sa méthode `preventDefault()`.

Certains comportements par défaut ne peuvent pas être évités. Par exemple, Flash Player et AIR génèrent un objet MouseEvent lorsque l'utilisateur double-clique sur un mot dans un objet TextField. Le comportement par défaut, qui ne peut être évité, consiste à mettre en évidence le mot situé sous le curseur.

De nombreux types d'objets événement ne sont associés à aucun comportement par défaut. Par exemple, l'objet événement Connect, que Flash Player distribue lorsqu'une connexion réseau est établie, n'est associé à aucun comportement par défaut. La documentation de l'API relative à la classe Event et ses sous-classes fait l'inventaire de chaque type d'événement, décrit le comportement par défaut qui lui est éventuellement associé et indique si ce dernier peut être évité.

Il est important de comprendre que les comportements par défaut sont uniquement associés à des objets événements distribués par Flash Player ou AIR ; il n'en existe aucun pour les objets événements distribués via ActionScript par programmation. Par exemple, vous pouvez utiliser les méthodes de la classe EventDispatcher pour distribuer un objet événement du type `textInput`, mais cet objet ne sera associé à aucun comportement par défaut. En d'autres termes, Flash Player et AIR n'affichent aucun caractère dans un objet TextField en réponse à un événement `textInput` que vous avez distribué par programmation.

Nouveautés des écouteurs d'événement dans ActionScript 3.0

Pour les développeurs qui connaissent bien la méthode ActionScript 2.0 `addListener()`, il peut être utile de souligner les différences entre le modèle d'écouteur d'événement d'ActionScript 2.0 et le modèle d'événements d'ActionScript 3.0. La liste ci-après décrit les principales différences entre ces deux modèles d'événements :

- Pour ajouter des écouteurs d'événement dans ActionScript 2.0, vous utilisez, selon le cas, `addListener()` ou `addEventListener()`. Dans ActionScript 3.0, il faut utiliser `addEventListener()` dans tous les cas.
- ActionScript 2.0 ne propose aucun flux d'événements dans ActionScript 2.0, ce qui signifie que la méthode `addListener()` peut uniquement être appelée sur l'objet qui émet l'événement. Dans ActionScript 3.0, la méthode `addEventListener()` peut être appelée sur tout objet faisant partie du flux d'événements.
- Dans ActionScript 2.0, les écouteurs d'événement peuvent être des fonctions, des méthodes ou des objets, alors que dans ActionScript 3.0, seules les fonctions et les méthodes peuvent agir comme écouteurs d'événement.

Flux d'événements

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

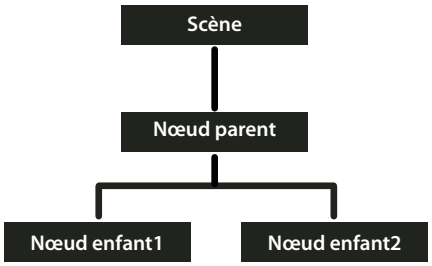
Flash Player ou AIR distribue des objets événements dès que survient un événement. Si la cible d'événement ne se trouve pas dans la liste d'affichage, Flash Player ou AIR distribue l'objet événement directement à la cible. Par exemple, Flash Player distribue l'objet événement Progress directement à un objet URLStream. Cependant, si la cible d'événement se trouve dans la liste d'affichage, Flash Player distribue l'objet événement à la liste d'affichage, dans laquelle l'objet chemine jusqu'à atteindre la cible d'événement.

Le *flux d'événements* représente le parcours que suivra un objet événement dans la liste d'affichage. Cette liste s'organise de manière hiérarchique, pour constituer une arborescence. Au sommet de la liste d'affichage se trouve la scène, un conteneur d'objet d'affichage spécial qui lui sert de racine. La Scène, représentée par la classe flash.display.Stage, est uniquement accessible via un objet d'affichage. Chaque objet d'affichage présente une propriété appelée `stage`, qui renvoie à la scène de cette application.

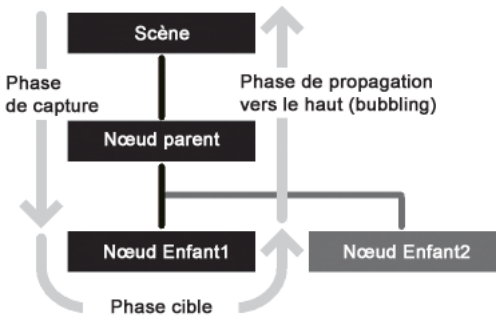
Lorsque Flash Player ou AIR distribue un objet d'événement pour un événement associé à une liste d'affichage, celui-ci effectue un aller-retour entre la Scène et le *nœud cible*. Selon la définition de la spécification d'événements DOM, le nœud cible est le nœud qui représente la cible d'événement. En d'autres termes, le noeud cible est l'objet de la liste d'affichage au niveau duquel est survenu l'événement. Par exemple, si l'utilisateur clique sur un objet de la liste d'affichage appelé `child1`, Flash Player ou AIR distribue un objet événement dont le noeud cible est `child1`.

Le flux d'événements se décompose en trois phases. La première correspond à la phase de capture, qui comprend tous les noeuds de la Scène jusqu'au parent du noeud cible. La deuxième partie est appelée la phase cible, qui comprend uniquement le noeud cible. La troisième partie s'appelle la phase de propagation. Elle comprend les nœuds rencontrés lors du cheminement du parent du nœud cible jusqu'à la scène.

Le nom de ces phases prend tout son sens si vous envisagez la liste d'affichage comme une hiérarchie verticale dont le sommet est la Scène, comme illustré par le schéma suivant :



Si un utilisateur clique sur `Child1 Node`, Flash Player ou AIR distribue un objet événement dans ce flux d'événements. Comme le montre l'illustration suivante, le parcours de l'objet commence à `Scène`. L'objet descend ensuite jusqu'à `Nœud parent`, puis vers `Nœud enfant1`. Il se propage alors vers le haut jusqu'à `Scène`, en repassant par `Nœud parent` pour rejoindre `Scène`.



Dans cet exemple, la phase de capture comprend `Scène` et `Nœud parent` pendant le trajet descendant initial. La phase cible comprend le temps passé au nœud `Nœud enfant1`. La phase de propagation comprend les nœuds `Nœud parent` et `Scène`, qui se trouvent sur le chemin du retour vers le nœud racine.

Le flux d'événements contribue au renforcement du système de gestion des événements par rapport aux versions précédentes d'ActionScript. Dans ces dernières, le flux d'événements est inexistant, ce qui signifie que les écouteurs d'événement s'ajoutent uniquement à l'objet qui génère l'événement. Dans ActionScript 3.0, vous pouvez ajouter des écouteurs d'événement aussi bien à un noeud cible qu'à tout autre noeud du flux d'événements.

Cette possibilité d'ajouter des écouteurs d'événement tout au long du flux d'événements s'avère particulièrement utile lorsqu'un composant d'interface comprend plusieurs objets. Par exemple, un objet bouton contient souvent un objet texte qui sert de libellé au bouton. Sans la possibilité d'ajouter un écouteur au flux d'événements, il faudrait en ajouter un à l'objet bouton et un à l'objet texte pour être sûr d'être averti des événements de clic survenant à tout endroit du bouton. Le flux d'événements vous permet, au contraire, de placer un seul écouteur d'événement sur l'objet bouton afin de gérer les événements de clic, qu'ils se produisent sur l'objet texte ou sur des zones de l'objet bouton non couvertes par l'objet texte.

Cependant, certains objets événements ne participent pas aux trois phases du flux d'événements. Certains types d'événements, tels que `enterFrame` et `init`, sont distribués directement au nœud cible et ne participent ni à la phase de capture, ni à la phase de propagation. D'autres événements peuvent cibler des objets qui ne font pas partie de la liste d'affichage, par exemple les événements distribués à une occurrence de la classe Socket. Ces objets événements aboutissent directement à l'objet cible, sans participer à la phase de capture et de propagation.

Pour savoir comment se comporte un type d'événement particulier, vous pouvez consulter la documentation de l'API ou examiner les propriétés de l'objet événement. Cette dernière méthode est décrite à la section suivante.

Objets événement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les objets événements jouent deux rôles essentiels dans le nouveau système de gestion des événements. Tout d'abord, ces objets représentent de véritables événements, puisqu'ils stockent dans un ensemble de propriétés des informations relatives à des événements précis. Ils contiennent en outre un jeu de méthodes qui vous permet de manipuler les objets événement et d'agir sur le comportement du système de gestion des événements.

Pour faciliter l'accès à ces propriétés et ces méthodes, l'API Flash Player définit une classe Event qui constitue la classe de base de tous les objets événements. La classe Event définit un jeu fondamental de propriétés et de méthodes commun à tous les objets événement.

Cette section commence par étudier les propriétés de la classe Event avant de décrire les méthodes de cette même classe, puis explique l'existence de sous-classes dans la classe Event.

Présentation des propriétés de la classe Event

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Event définit plusieurs propriétés et constantes en lecture seule qui fournissent des informations essentielles sur l'objet événement. Les points suivants revêtent une importance particulière :

- Les types d'objet événement sont représentés par des constantes et stockés dans la propriété `Event.type`.
- La possibilité d'éviter le comportement par défaut d'un événement est représentée par une valeur booléenne, stockée dans la propriété `Event.cancelable`.
- Les informations relatives au flux d'événements se trouvent dans les propriétés restantes.

Types d'objets événement

Chaque objet événement est associé à un type d'événement. Les types d'événements sont stockés dans la propriété `Event.type` sous forme de chaîne. Il est utile de connaître le type d'un objet événement car votre code peut alors distinguer les objets de types différents. Par exemple, le code suivant spécifie que la fonction `clickHandler()` doit répondre à tous les objets événements clic de souris transmis à `myDisplayObject` :

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

La classe Event est elle-même associée à deux douzaines de types d'événement, représentés par des constantes de la classe Event. Dans cet extrait de la définition de la classe Event, certaines de ces constantes sont illustrées :

```
package flash.events
{
    public class Event
    {
        // class constants
        public static const ACTIVATE:String = "activate";
        public static const ADDED:String= "added";
        // remaining constants omitted for brevity
    }
}
```

Ces constantes permettent de faire facilement référence à des types d'événement précis. Vous devez utiliser ces constantes au lieu des chaînes qu'elles représentent. Si vous orthographiez de manière incorrecte un nom de constante dans votre code, le compilateur peut détecter l'erreur. Si vous utilisez les chaînes qu'elles représentent, une erreur de frappe ne sera pas forcément détectée lors de la compilation et pourrait provoquer un comportement inattendu, difficile à déboguer. Par exemple, utilisez le code suivant pour ajouter un écouteur d'événement :

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

plutôt que :

```
myDisplayObject.addEventListener("click", clickHandler);
```

Informations relatives aux comportements par défaut

Le code que vous écrivez est en mesure de vérifier si le comportement par défaut d'un objet événement donné peut être évité. Pour ce faire, il doit accéder à la propriété `cancelable`. La propriété `cancelable` contient une valeur booléenne qui indique si le comportement par défaut peut être évité ou non. Vous pouvez éviter, ou annuler, le comportement par défaut de quelques événements à l'aide de la méthode `preventDefault()`. Pour plus d'informations, voir Annulation d'un comportement associé par défaut à un événement sous « Présentation des méthodes de la classe Event » à la page 137.

Informations de flux d'événements

Les propriétés restantes de la classe Event contiennent des informations importantes sur l'objet événement et ses relations au flux d'événements, comme l'explique la liste suivante :

- La propriété `bubbles` contient des informations sur les parties du flux d'événements auquel participe l'objet événement.
- La propriété `eventPhase` indique la phase actuelle du flux d'événements.
- La propriété `target` stocke une référence à la cible d'événement.
- La propriété `currentTarget` stocke une référence de l'objet de liste d'affichage qui traite actuellement l'objet événement.

La propriété bubbles

On dit d'un événement qu'il se propage lorsqu'il participe à la phase de propagation du flux d'événements, c'est-à-dire quand l'objet événement est transmis du nœud cible via ses ascendants jusqu'à la Scène. La propriété `Event.bubbles` stocke une valeur booléenne qui indique si l'objet événement participe à la phase de propagation. Tous les événements qui se propagent vers le haut participent également aux phases de capture et cible ; de tels événements participent donc aux trois phases du flux d'événements. Si la valeur est `true`, l'objet événement participe aux trois phrases. Si la valeur est `false`, l'objet événement ne participe pas à la phase de propagation.

La propriété eventPhase

Vous pouvez déterminer la phase d'événement de tout objet événement grâce à sa propriété `eventPhase`. La propriété `eventPhase` a pour valeur un entier non signé qui représente l'une des trois phases du flux d'événements. L'API de Flash Player définit une classe EventPhase distincte qui contient trois constantes correspondant aux trois valeurs entières non signées, comme illustré par l'extrait de code suivant :

```
package flash.events
{
    public final class EventPhase
    {
        public static const CAPTURING_PHASE:uint = 1;
        public static const AT_TARGET:uint = 2;
        public static const BUBBLING_PHASE:uint= 3;
    }
}
```

Ces constantes correspondent aux trois valeurs valables pour la propriété `eventPhase`. Vous pouvez utiliser ces constantes pour améliorer la lisibilité de votre code. Supposons par exemple que vous souhaitiez être sûr qu'une fonction appelée `myFunc()` soit uniquement appelée lorsque la cible d'événement se trouve dans la scène cible. Le code suivant vous permet de tester cette condition :

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

La propriété target

La propriété `target` contient une référence à l'objet cible de l'événement. Dans certains cas, ce système est simple, par exemple, lorsqu'un micro devient actif, la cible de l'objet événement est l'objet Microphone. Toutefois, si la cible se trouve sur la liste d'affichage, il faut tenir compte de la hiérarchie de cette dernière. Par exemple, si un utilisateur clique avec la souris sur un point correspondant à plusieurs objets de la liste d'affichage qui se chevauchent, Flash Player et AIR choisissent toujours comme cible d'événement l'objet qui se trouve le plus loin de la Scène.

Dans des fichiers SWF complexes, et particulièrement ceux dont les boutons sont régulièrement ornés d'objets enfant plus petits, la propriété `target` ne doit pas être utilisée fréquemment car elle pointera souvent vers l'objet enfant du bouton plutôt que vers le bouton lui-même. Dans de telles situations, il est courant d'ajouter des écouteurs d'événement au bouton et d'utiliser la propriété `currentTarget`. En effet, cette dernière pointe vers le bouton alors que la propriété `target` peut pointer vers l'un des enfants du bouton.

La propriété currentTarget

La propriété `currentTarget` contient une référence de l'objet de liste d'affichage qui traite actuellement l'objet événement. Même s'il peut paraître étrange de ne pas savoir quel nœud traite actuellement l'objet événement que vous étudiez, gardez à l'esprit que vous pouvez ajouter une fonction écouteur à n'importe quel objet d'affichage du flux d'événements de l'objet événement en question. En outre, cette fonction écouteur peut être placée à tout endroit. Par ailleurs, la même fonction écouteur peut être ajoutée à différents objets d'affichage. L'utilité de la propriété `currentTarget` augmente donc avec la taille et la complexité du projet.

Présentation des méthodes de la classe Event

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Event contient trois catégories de méthodes :

- Les méthodes d'utilitaire, qui peuvent créer des copies d'un objet événement ou le convertir en chaîne
- Les méthodes de flux d'événements, qui suppriment les objets événements du flux d'événements
- Les méthodes de comportement par défaut, qui annulent le comportement par défaut ou vérifient s'il a été annulé

Méthodes d'utilitaire de la classe Event

La classe Event compte deux méthodes d'utilitaire. La méthode `clone()` permet de créer des copies d'un objet événement. La méthode `toString()` permet de représenter sous forme de chaînes les propriétés d'un objet événement ainsi que leurs valeurs. Bien qu'utilisées en interne par le modèle d'événements, ces deux méthodes sont mises à la disposition des développeurs pour un usage générique.

Pour les développeurs expérimentés qui souhaitent créer des sous-classes de la classe Event, il est nécessaire de redéfinir et d'implémenter des versions de ces deux méthodes d'utilitaires afin de garantir le bon fonctionnement de la sous-classe d'événement.

Arrêt du flux d'événements

La méthode `Event.stopPropagation()` ou `Event.stopImmediatePropagation()` vous permet d'arrêter le cheminement d'un objet événement dans le flux d'événements. Quasi identiques, ces deux méthodes diffèrent uniquement en ce que les autres écouteurs d'événement du nœud actuel sont autorisés ou non à s'exécuter :

- La méthode `Event.stopPropagation()` empêche l'objet événement de passer au nœud suivant mais seulement après que tous les autres écouteurs du nœud actuel ont été autorisés à s'exécuter.
- La méthode `Event.stopImmediatePropagation()` empêche l'objet événement de passer au nœud suivant sans autoriser les autres écouteurs du nœud actuel à s'exécuter.

Quelle que soit la méthode appelée, elle n'a aucun effet sur la réalisation du comportement par défaut de l'événement. Utilisez les méthodes de comportement par défaut de la classe Event pour éviter le comportement par défaut.

Annulation d'un comportement associé par défaut à un événement

Deux méthodes sont associées à l'annulation du comportement par défaut : `preventDefault()` et `isDefaultPrevented()`. Appelez la méthode `preventDefault()` pour annuler le comportement associé par défaut à un événement. Pour vérifier si `preventDefault()` a déjà été appelée sur un objet événement, appelez la méthode `isDefaultPrevented()`, qui renvoie la valeur `true` si la méthode a déjà été appelée, `false` dans le cas contraire.

La méthode `preventDefault()` fonctionne uniquement s'il est possible d'annuler le comportement par défaut de l'événement. Pour vérifier que c'est le cas, reportez-vous à la documentation de l'API de ce type d'événement ou examinez la propriété `cancelable` de l'objet événement à l'aide du code ActionScript.

L'annulation du comportement par défaut n'a aucun effet sur la progression d'un objet événement dans le flux d'événements. Utilisez les méthodes de flux d'événements de la classe Event pour supprimer un objet événement du flux d'événements.

Sous-classes de la classe Event

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour de nombreux événements, le jeu de propriétés commun, défini dans la classe Event est suffisant. Néanmoins, d'autres événements présentent des caractéristiques exclusives qui ne peuvent être capturées par les propriétés disponibles dans la classe Event. Pour ces événements, ActionScript 3.0 définit plusieurs sous-classes de la classe Evénement.

Chaque sous-classe fournit d'autres propriétés et types d'événements propres à la catégorie d'événement considérée. Par exemple, les événements liés aux actions de la souris présentent plusieurs caractéristiques uniques, que les propriétés définies dans la classe Event ne peuvent capturer. La classe MouseEvent constitue une extension de la classe Event puisqu'elle ajoute dix propriétés contenant des informations telles que l'emplacement de l'événement de souris et les éventuelles touches actionnées en même temps.

Une sous-classe d'Event contient également des constantes qui représentent de types d'événement associés à la sous-classe. Par exemple, la classe MouseEvent définit des constantes pour plusieurs types d'événement de souris, notamment `click`, `doubleClick`, `mouseDown` et `mouseUp`.

Comme le décrit la section consacrée aux méthodes d'utilitaire de la classe Event dans « Objets événement » à la page 135, lors de la création d'une sous-classe d'Event, vous devez bloquer les méthodes `clone()` et `toString()` pour fournir la fonctionnalité propre à la sous-classe.

Ecouteurs d'événement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les écouteurs d'événement, également appelés gestionnaires d'événements, sont des fonctions que Flash Player et AIR exécutent en réponse à des événements déterminés. La procédure d'ajout d'un écouteur d'événement se déroule en deux temps. En premier lieu, vous créez une fonction ou méthode de classe que Flash Player ou AIR doit exécuter en réponse à l'événement. On parle parfois de fonction d'écouteur ou de fonction de gestionnaire d'événement. En second lieu, vous utilisez la méthode `addEventListener()` pour enregistrer la fonction d'écouteur auprès de la cible de l'événement ou tout autre objet de la liste d'affichage qui appartient au flux d'événements approprié.

Création d'une fonction d'écouteur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La création d'une fonction d'écouteur est un domaine dans lequel le modèle d'événements ActionScript 3.0 diffère du modèle d'événements DOM. Dans le modèle d'événements DOM, on distingue clairement un écouteur d'événement et une fonction d'écouteur : un écouteur d'événement est une occurrence de classe qui implémente l'interface EventListener, tandis qu'une fonction d'écouteur est une méthode de cette classe appelée `handleEvent()`. Dans le modèle d'événements DOM, vous enregistrez l'occurrence de classe qui contient la fonction d'écouteur, plutôt que la fonction d'écouteur elle-même.

Le modèle d'événements ActionScript ne fait aucune distinction entre l'écouteur d'événement et la fonction d'écouteur. L'interface EventListener est inexistante dans ActionScript 3.0 et les fonctions d'écouteur peuvent être définies en dehors de toute classe ou au sein d'une classe. Par ailleurs, il n'est pas nécessaire de nommer les fonctions d'écouteur `handleEvent()` ; vous pouvez utiliser tout identifiant valable. Dans ActionScript 3.0, vous enregistrez le nom de la fonction d'écouteur elle-même.

Fonction d'écouteur définie en dehors de toute classe

Le code suivant crée un fichier SWF simple qui affiche une forme carrée de couleur rouge. Une fonction d'écouteur appelée `clickHandler()`, qui n'appartient à aucune classe, écoute les événements de clic de souris dans le carré rouge.

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, clickHandler);
    }
}

function clickHandler(event:MouseEvent):void
{
    trace("clickHandler detected an event of type: " + event.type);
    trace("the this keyword refers to: " + this);
}
```

Lorsqu'un utilisateur interagit avec le fichier SWF résultant, en cliquant sur le carré, Flash Player ou AIR génère la sortie de suivi ci-après :

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Notez que l'objet événement est transmis sous forme d'instruction à `clickHandler()`. Cela permet à votre fonction d'écouteur d'examiner l'objet événement. Dans cet exemple, vous utilisez la propriété `type` de l'objet événement pour vérifier que cet événement correspond à un clic.

L'exemple vérifie aussi la valeur du mot-clé `this`. Dans ce cas, `this` représente l'objet global, ce qui est logique puisque la fonction est définie en dehors de toute classe ou objet personnalisé.

Fonction d'écouteur définie comme méthode de classe

L'exemple ci-dessous est identique au précédent, qui définit la classe ClickExample, sauf que la fonction `clickHandler()` est définie comme méthode de la classe ChildSprite :

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, clickHandler);
    }
    private function clickHandler(event:MouseEvent):void
    {
        trace("clickHandler detected an event of type: " + event.type);
        trace("the this keyword refers to: " + this);
    }
}
```

Lorsqu'un utilisateur interagit avec le fichier SWF résultant, en cliquant sur le carré rouge, Flash Player ou AIR génère la sortie de suivi ci-après :

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

Notez que le mot-clé `this` renvoie à l'occurrence de ChildSprite appelée `child`. Voici un changement de comportement par rapport à ActionScript 2.0. Si vous utilisiez des composants dans ActionScript 2.0, vous vous rappelez sans doute que lorsqu'une méthode de classe était transmise à `UIEventDispatcher.addEventListener()`, l'étendue de la méthode était liée au composant qui émettait l'événement, et non à la classe dans laquelle la méthode d'écouteur était définie. En d'autres termes, si vous utilisiez cette technique dans ActionScript 2.0, le mot-clé `this` renvoyait au composant émettant l'événement et non à l'occurrence de ChildSprite.

Pour certains développeurs, il s'agissait d'un vrai problème car cela signifiait qu'ils ne pouvaient accéder à aucune autre méthode et propriété de la classe qui contenait la méthode d'écouteur. Pour le contourner, les programmeurs d'ActionScript 2.0 pouvaient utiliser la classe `mx.util.Delegate` pour modifier l'étendue de la méthode d'écouteur. Cette manipulation n'est plus nécessaire puisque ActionScript 3.0 crée une méthode liée lorsque `addEventListener()` est appelée. Par conséquent, le mot-clé `this` fait référence à l'occurrence de ChildSprite appelée `child` et le programmeur peut accéder aux autres méthodes et propriétés de la classe ChildSprite.

Ecouteur d'événement à ne pas utiliser

Une troisième technique permet de créer un objet générique dont l'une des propriétés pointe vers une fonction d'écouteur affectée dynamiquement. Elle est cependant déconseillée. Nous l'évoquons ici en raison de son utilisation courante dans ActionScript 2.0 ; il n'est toutefois pas recommandé de l'utiliser dans ActionScript 3.0. Cette mise en garde tient au fait que le mot-clé `this` fera référence à l'objet global et non à l'objet écouteur.

L'exemple ci-après est identique à l'exemple précédent de la classe ClickExample, sauf que la fonction d'écouteur est définie comme faisant partie d'un objet générique appelé `myListenerObj` :

```
package
{
    import flash.display.Sprite;

    public class ClickExample extends Sprite
    {
        public function ClickExample()
        {
            var child:ChildSprite = new ChildSprite();
            addChild(child);
        }
    }
}

import flash.display.Sprite;
import flash.events.MouseEvent;

class ChildSprite extends Sprite
{
    public function ChildSprite()
    {
        graphics.beginFill(0xFF0000);
        graphics.drawRect(0,0,100,100);
        graphics.endFill();
        addEventListener(MouseEvent.CLICK, myListenerObj.clickHandler);
    }
}

var myListenerObj:Object = new Object();
myListenerObj.clickHandler = function (event:MouseEvent):void
{
        trace("clickHandler detected an event of type: " + event.type);
        trace("the this keyword refers to: " + this);
}
```

Les résultats de trace seront les suivants :

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

On s'attendrait à ce que `this` fasse référence à `myListenerObj` et que la sortie de suivi soit `[object Object]`, mais le mot-clé renvoie en fait à l'objet global. Lorsque vous transmettez un nom de propriété dynamique comme instruction à `addEventListener()`, Flash Player ou AIR est incapable de créer une méthode liée. En effet, ce que vous transmettez comme paramètre `listener` n'est rien de plus que l'adresse mémoire de votre fonction d'écouteur ; Flash Player et AIR n'ont aucun moyen de lier cette adresse à l'occurrence de `myListenerObj` .

Gestion des écouteurs d'événement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez gérer vos fonctions d'écouteur à l'aide des méthodes de l'interface IEventDispatcher. Cette interface est la version ActionScript 3.0 de l'interface EventTarget du modèle d'événements DOM. Bien que le nom IEventDispatcher semble impliquer que l'objet principal de la classe est l'envoi (ou la distribution) des objets événements, les méthodes qui lui correspondent servent en fait plus souvent à l'enregistrement, la vérification et la suppression des écouteurs d'événement. L'interface IEventDispatcher définit cinq méthodes, comme illustré dans le code suivant :

```
package flash.events
{
    public interface IEventDispatcher
    {
        function addEventListener(eventName:String,
                        listener:Object,
                        useCapture:Boolean=false,
                        priority:Integer=0,
                        useWeakReference:Boolean=false):Boolean;

        function removeEventListener(eventName:String,
                    listener:Object,
                    useCapture:Boolean=false):Boolean;

        function dispatchEvent(eventObject:Event):Boolean;

        function hasEventListener(eventName:String):Boolean;
        function willTrigger(eventName:String):Boolean;
    }
}
```

L'API de Flash Player implémente l'interface IEventDispatcher à l'aide de la classe Event Dispatcher. Cette dernière constitue la classe de base de toutes les classes pouvant servir de cibles d'événement ou faire partie d'un flux d'événements. Par exemple, la classe DisplayObject hérite de la classe EventDispatcher, par conséquent, tout objet de la liste d'affichage peut accéder aux méthodes de l'interface IEventDispatcher.

Ajout des écouteurs d'événement

La méthode `addEventListener()` est la clé de voûte de l'interface IEventDispatcher. Elle permet d'enregistrer les fonctions d'écouteurs. Les deux paramètres requis sont `type` et `listener`. Le paramètre `type` spécifie le type d'événement. Avec le paramètre `listener`, vous pouvez spécifier la fonction d'écouteur qui doit s'exécuter lorsque l'événement survient. Le paramètre `listener` peut être une référence à une fonction ou une méthode de classe.

n'utilisez pas de parenthèses pour stipuler le paramètre `listener`. Par exemple, la fonction `clickHandler()` est spécifiée sans parenthèses dans l'appel suivant à la méthode `addEventListener()` :

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

Le paramètre `useCapture` de la méthode `addEventListener()` vous permet de contrôler la phase du flux d'événements pendant laquelle votre écouteur sera actif. Si `useCapture` a la valeur `true`, votre écouteur sera actif pendant la phase de capture du flux d'événements. Si `useCapture` a la valeur `false`, votre écouteur sera actif pendant la phase cible et la phase de propagation du flux d'événements. Pour écouter un événement pendant toutes les phases du flux d'événements, vous devez appeler deux fois `addEventListener()` ; la première fois, `useCapture` prend la valeur `true`, la seconde fois, `useCapture` prend la valeur `false`.

Le paramètre `priority` de la méthode `addEventListener()` ne fait pas officiellement partie du modèle d'événements DOM de niveau 3. Il est inclus dans ActionScript 3.0 pour vous offrir une plus grande souplesse dans l'organisation de vos écouteurs d'événement. Lorsque vous appelez `addEventListener()`, vous pouvez définir la priorité de cet écouteur d'événement en transmettant une valeur entière comme paramètre `priority`. La valeur par défaut est 0. Vous pouvez toutefois utiliser une valeur entière négative ou positive. Plus le nombre est élevé, plus l'exécution de l'écouteur d'événement est rapide. Les écouteurs d'événement de priorité équivalente sont exécutés suivant l'ordre dans lequel ils ont été ajoutés : plus l'écouteur est ajouté tôt, plus il est exécuté rapidement.

Le paramètre `useWeakReference` vous permet de spécifier si la référence à la fonction d'écouteur est faible ou normale. En lui attribuant la valeur `true`, vous évitez les situations dans lesquelles les fonctions d'écouteurs demeurent dans la mémoire alors qu'elles sont inutiles. Flash Player et AIR utilisent une technique appelée *nettoyage* pour effacer de la mémoire les objets qui ne servent plus. Un objet est considéré comme inutilisé lorsqu'il n'apparaît dans aucune référence. Le nettoyeur de mémoire ignore les références faibles, c'est-à-dire qu'une fonction d'écouteur vers laquelle pointe uniquement une référence faible est incluse dans le nettoyage.

Suppression des écouteurs d'événement

La méthode `removeEventListener()` permet de supprimer un écouteur d'événement dont vous n'avez plus besoin. Il est judicieux de supprimer tous les écouteurs qui ne seront plus utilisés. Les paramètres requis sont notamment `eventName` et `listener`, soit les mêmes que ceux requis pour la méthode `addEventListener()`. Rappel : pour écouter les événements pendant toutes les phases du flux d'événements, vous pouvez appeler `addEventListener()` deux fois, en attribuant à `useCapture` la valeur `true` la première, puis `false` la seconde. Pour supprimer les deux écouteurs d'événement, il serait nécessaire d'appeler `removeEventListener()` à deux reprises, la première fois en attribuant la valeur `true` à `useCapture`, la seconde fois en utilisant la valeur `false`.

Distribution d'événements

La méthode `dispatchEvent()` peut servir aux développeurs chevronnés pour distribuer un objet événement personnalisé dans le flux d'événements. Cette méthode accepte un seul paramètre, une référence à l'objet événement, qui doit être une occurrence de la classe Event ou de l'une de ces sous-classes. Après distribution, la propriété `target` de l'objet événement est définie avec l'objet sur lequel portait l'appel `dispatchEvent()`.

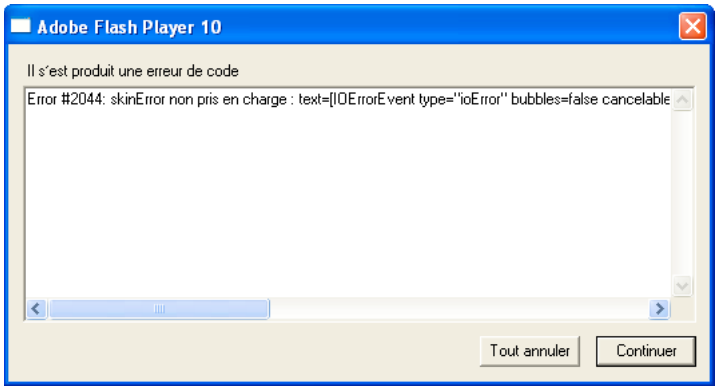
Vérification des écouteurs d'événement existants

Les deux dernières méthodes de l'interface IEventDispatcher fournissent des informations précieuses sur l'existence des écouteurs d'événement. La méthode `hasEventListener()` renvoie la valeur `true` si un écouteur d'événement est détecté pour un type d'événement spécifique sur un objet particulier de la liste d'affichage. La méthode `willTrigger()` renvoie également la valeur `true` si un écouteur est détecté pour un objet donné de la liste d'affichage. Cependant `willTrigger()` vérifie les écouteurs sur l'objet d'affichage en question mais également sur tous les ascendants de cet objet dans l'ensemble des phases du flux d'événements.

Evénements d'erreur sans écouteurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Plus que les événements, les exceptions constituent le mécanisme principal de gestion des erreurs dans ActionScript 3.0. Toutefois, la gestion des exceptions ne fonctionne pas sur les opérations asynchrones telles que les chargements de fichiers. Si une erreur survient pendant une opération asynchrone, Flash Player et AIR distribuent un objet événement d'erreur. Si vous ne créez pas d'écouteur pour l'événement d'erreur, les versions de débogage de Flash Player et AIR affichent une boîte de dialogue comportant des informations sur l'erreur en question. Par exemple, la version de débogage de Flash Player affiche la boîte de dialogue suivante, qui décrit l'erreur associée à une tentative de chargement d'un fichier par l'application à partir d'une URL non valide :



La plupart des événements d'erreur reposent sur la classe ErrorEvent. Ils présentent donc une propriété appelée `text`, qui sert au stockage du message d'erreur que Flash Player ou AIR affiche. Il existe deux exceptions : les classes StatusEvent et NetStatusEvent. Ces deux classes possèdent une propriété `level` (`StatusEvent.level` et `NetStatusEvent.info.level`). Lorsque la valeur de la propriété `level` est `error`, ces types d'événement sont considérés comme des événements d'erreur.

Un événement d'erreur n'interrompt pas l'exécution du fichier SWF. Il se traduit uniquement par l'affichage d'une boîte de dialogue dans les versions de débogage des navigateurs et des lecteurs autonomes, d'un message dans le panneau de sortie du lecteur de création et d'une entrée dans le fichier journal d'Adobe Flash Builder. Aucune manifestation n'est visible dans les autres versions de Flash Player ou AIR.

Exemple de gestion des événements : Alarm Clock

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple Alarm Clock correspond à une horloge qui permet à l'utilisateur de déterminer l'heure à laquelle l'alarme doit se déclencher et d'afficher un message en même temps. Il repose sur l'application SimpleClock du chapitre « Utilisation des dates et des heures » à la page 1 et illustre de nombreux aspects de l'utilisation des événements dans ActionScript 3.0, notamment les suivants :

- Ecoute des événements et réponse
- Notification d'un événement aux écouteurs
- Créer un type d'événement personnalisé

Pour obtenir les fichiers d'application Flash Professional associés à cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Pour obtenir les fichiers d'application Flex associés à cet exemple, voir http://www.adobe.com/go/as3examples_fr. Les fichiers d'application Alarm Clock se trouvent dans le dossier Samples/AlarmClock. Il s'agit des fichiers suivants :

Fichier	Description
AlarmClockApp.mxml ou AlarmClockApp.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/clock/AlarmClock.as	Classe permettant d'étendre la classe SimpleClock, qui ajoute la fonctionnalité de réveil.
com/example/programmingas3/clock/AlarmEvent.as	Une classe d'événement personnalisé (sous-classe de flash.events.Event), qui sert d'objet événement à l'événement `alarm` de la classe AlarmClock.
com/example/programmingas3/clock/AnalogClockFace.as	Dessine une horloge ronde et les aiguilles des heures, des minutes et des secondes en fonction de l'heure (décrit dans l'exemple SimpleClock).
com/example/programmingas3/clock/SimpleClock.as	Composant d'interface d'horloge doté d'une fonctionnalité simple de mesure temporelle (décrit dans l'exemple SimpleClock).

Présentation du réveil

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans cet exemple, la principale fonctionnalité de l'horloge (dont la mesure du temps et l'affichage du cadran) réutilise le code de l'application SimpleClock, décrite à la section « Exemple de date et heure : horloge analogique simple » à la page 6. La classe AlarmClock étend la classe SimpleClock de cet exemple en y ajoutant la fonctionnalité de réveil requise : réglage de l'heure de déclenchement et avertissement une fois l'alarme déclenchée.

Le rôle des événements est de fournir un avertissement lorsque se produit quelque chose. La classe AlarmClock expose l'événement Alarme, à l'écoute duquel d'autres objets peuvent être placés afin d'effectuer les actions voulues. En outre, la classe AlarmClock utilise une occurrence de la classe Timer pour déterminer à quel moment déclencher l'alarme. Comme la classe AlarmClock, la classe Timer fournit un événement pour avertir d'autres objets (une occurrence de AlarmClock dans ce cas) une fois un certain délai écoulé. Comme dans la plupart des applications ActionScript, les événements constituent une part importante de la fonctionnalité de l'exemple Alarm Clock.

Déclenchement de l'alarme

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme mentionné plus haut, la seule fonctionnalité de la classe AlarmClock est liée à la définition et au déclenchement de l'alarme. La classe intégrée Timer (flash.utils.Timer) permet au développeur de définir du code qui sera exécuté après un délai spécifique. La classe AlarmClock utilise une occurrence de Timer pour déterminer le moment auquel déclencher l'alarme.

```
import flash.events.TimerEvent;
import flash.utils.Timer;

/**
 * The Timer that will be used for the alarm.
 */
public var alarmTimer:Timer;
...
/**
 * Instantiates a new AlarmClock of a given size.
 */
public override function initClock(faceSize:Number = 200):void
{
    super.initClock(faceSize);
    alarmTimer = new Timer(0, 1);
    alarmTimer.addEventListener(TimerEvent.TIMER, onAlarm);
}
```

L'occurrence de Timer définie dans la classe AlarmClock est appelée `alarmTimer`. La méthode `initClock()`, qui effectue les opérations de configuration nécessaires à l'occurrence de AlarmClock, exploite la variable `alarmTimer` de deux manières. Tout d'abord, la variable est instanciée avec les paramètres indiquant à l'occurrence de Timer d'attendre 0 milliseconde et de déclencher l'événement timer une seule fois. Après instanciation de `alarmTimer`, le code appelle la méthode `addEventListener()` de cette variable pour indiquer qu'il veut écouter l'événement `timer` de cette variable. Le fonctionnement d'une occurrence de Timer repose sur la distribution de l'événement `timer` après un certain délai. La classe AlarmClock doit savoir quand l'événement `timer` est distribué afin de déclencher sa propre alarme. En appelant `addEventListener()`, le code AlarmClock s'enregistre comme écouteur auprès de `alarmTimer`. Les deux paramètres indiquent que la classe AlarmClock souhaite écouter l'événement `timer` (indiqué par la constante `TimerEvent.TIMER`), et que lorsque l'événement survient, la méthode `onAlarm()` de la classe AlarmClock doit être appelée en réponse à l'événement.

Pour effectivement définir l'alarme, la méthode `setAlarm()` de la classe AlarmClock est appelée, comme suit :

```
/**
 * Sets the time at which the alarm should go off.
 * @param hour The hour portion of the alarm time.
 * @param minutes The minutes portion of the alarm time.
 * @param message The message to display when the alarm goes off.
 * @return The time at which the alarm will go off.
 */
public function setAlarm(hour:Number = 0, minutes:Number = 0, message:String = "Alarm!"):Date
{
    this.alarmMessage = message;
    var now:Date = new Date();
    // Create this time on today's date.
    alarmTime = new Date(now.fullYear, now.month, now.date, hour, minutes);

    // Determine if the specified time has already passed today.
    if (alarmTime <= now)
    {
        alarmTime.setTime(alarmTime.time + MILLISECONDS_PER_DAY);
    }

    // Stop the alarm timer if it's currently set.
    alarmTimer.reset();
    // Calculate how many milliseconds should pass before the alarm should
    // go off (the difference between the alarm time and now) and set that
    // value as the delay for the alarm timer.
    alarmTimer.delay = Math.max(1000, alarmTime.time - now.time);
    alarmTimer.start();

    return alarmTime;
}
```

Cette méthode effectue plusieurs opérations, notamment le stockage du message d'alarme et la création d'un objet Date (`alarmTime`) représentant le moment réel où l'alarme se déclenchera. Point le plus important de cette étude, le minuteur de la variable `alarmTimer`, dans les dernières lignes la méthode, est défini et activé. Tout d'abord, la méthode `reset()` est appelée, qui arrête le minuteur et le remet à zéro s'il est déjà reparti. Ensuite, l'heure actuelle (représentée par la variable `now`) est soustraite à la valeur de la variable `alarmTime` afin de déterminer combien de millisecondes doivent s'écouler avant le déclenchement de l'alarme. La classe Timer ne déclenche pas l'événement `timer` à une heure absolue ; c'est ce décalage relatif qui est attribué à la propriété `delay` d'`alarmTimer`. Enfin, la méthode `start()` est appelée pour lancer le minuteur.

Une fois le délai spécifié écoulé, `alarmTimer` distribue l'événement `timer`. Comme la classe AlarmClock s'est enregistrée comme écouteur auprès de sa méthode `onAlarm()` pour l'événement `timer`, lorsque celui-ci survient, `onAlarm()` est appelée.

```
/**
 * Called when the timer event is dispatched.
 */
public function onAlarm(event:TimerEvent):void
{
    trace("Alarm!");
    var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
    this.dispatchEvent(alarm);
}
```

Lorsqu'une méthode est enregistrée comme écouteur d'événement, elle doit être définie avec la signature adaptée (c'est-à-dire le jeu de paramètres et le type de renvoi de la méthode). Pour écouter l'événement `timer` de la classe Timer, une méthode doit comporter un paramètre dont le type de données est TimerEvent (flash.event.TimerEvent), une sous-classe de la classe Event. Lorsque l'occurrence de Timer appelle ses écouteurs d'événement, elle transmet une occurrence de TimerEvent à l'objet événement.

Notification de l'alarme à d'autres composants

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

De même que la classe Timer, la classe AlarmClock fournit un événement qui permet de transmettre des notifications à d'autres éléments de code lorsque l'alarme se déclenche. Pour qu'une classe puisse utiliser le système de gestion des événements intégré à ActionScript, elle doit implémenter l'interface flash.events.IEventDispatcher. La plupart du temps, cela se fait par extension de la classe flash.events.EventDispatcher, qui assure une implémentation standard de IEventDispatcher (ou par extension de l'une des sous-classes de EventDispatcher). Comme décrit précédemment, la classe AlarmClock étend la classe SimpleClock, qui (par le biais d'une chaîne d'héritage) étend la classe EventDispatcher. Ainsi, la classe AlarmClock intègre déjà une fonctionnalité lui permettant de fournir ses propres événements.

D'autres éléments de code peuvent s'enregistrer pour être notifiés de l'événement `alarm` de la classe AlarmClock en appelant la méthode `addEventListener()`, héritée de EventDispatcher. Lorsqu'une occurrence de AlarmClock est prête à notifier à d'autres éléments de code le déclenchement de l'événement `alarm`, elle le fait en appelant la méthode `dispatchEvent()`, également héritée de EventDispatcher.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

Ces lignes de code sont extraites de la méthode `onAlarm()` de la classe AlarmClock (présentée plus haut dans son intégralité). La méthode `dispatchEvent()` de l'occurrence de AlarmClock est appelée, puis elle notifie à tous les écouteurs enregistrés le déclenchement de l'événement `alarm` de l'occurrence de AlarmClock. Le paramètre transmis à `dispatchEvent()` est l'objet événement qui sera ensuite passé aux méthodes d'écouteur. Dans ce cas, il s'agit d'une occurrence de la classe AlarmEvent, une sous-classe de Event créée spécialement pour cet exemple.

Elaboration d'un événement d'alarme personnalisé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tous les écouteurs d'événement reçoivent un paramètre d'objet événement avec des informations sur l'événement qui a été déclenché. Dans bien des cas, l'objet événement est une occurrence de la classe Event. Dans d'autres cas néanmoins, il s'avère utile de fournir des informations complémentaires aux écouteurs d'événement. Il suffit pour cela de définir une nouvelle classe, sous-classe de la classe Event, et d'utiliser une occurrence de cette classe comme objet événement. Dans cet exemple, une occurrence de AlarmEvent est utilisée comme objet événement lorsque l'événement `alarm` de la classe AlarmClock est distribué. La classe AlarmEvent, présentée ici, fournit des informations complémentaires sur l'événement `alarm`, à savoir le message d'alarme :

```
import flash.events.Event;

/**
 * This custom Event class adds a message property to a basic Event.
 */
public class AlarmEvent extends Event
{
    /**
     * The name of the new AlarmEvent type.
     */
    public static const ALARM:String = "alarm";

    /**
     * A text message that can be passed to an event handler
     * with this event object.
     */
    public var message:String;

    /**
     *Constructor.
     *@param message The text to display when the alarm goes off.
     */
    public function AlarmEvent(message:String = "ALARM!")
    {
        super(ALARM);
        this.message = message;
    }
    ...
}
```

Le meilleur moyen de créer une classe d'objet événement personnalisée est de définir une classe qui étend la classEvent, comme illustré dans l'exemple précédent. Pour compléter la fonctionnalité héritée, la classe AlarmEvent définit une propriété `message` qui contient le texte du message d'alarme associé à l'événement. La valeur `message` est transmise sous forme de paramètre au constructeur AlarmEvent. La classe AlarmEvent définit également la constante `ALARM` qui peut servir à référencer l'événement (`alarm`) lors de l'appel de la méthode `addEventListener()` de la classe AlarmClock.

Outre l'ajout de fonctionnalité, chaque sous-classe Event doit redéfinir la méthode `clone()` héritée dans le cadre de la gestion des événements ActionScript. Les sous-classes Event peuvent éventuellement redéfinir la méthode `toString()` afin d'inclure les propriétés de l'événement personnalisé dans la valeur renvoyée par l'appel de la méthode `toString()`.

```
    /**
     * Creates and returns a copy of the current instance.
     * @return A copy of the current instance.
     */
    public override function clone():Event
    {
        return new AlarmEvent(message);
    }

    /**
     * Returns a String containing all the properties of the current
     * instance.
     * @return A string representation of the current instance.
     */
    public override function toString():String
    {
        return formatToString("AlarmEvent", "type", "bubbles", "cancelable", "eventPhase",
"message");
    }
```

La méthode `clone()` redéfinie doit renvoyer une nouvelle occurrence de la sous-classe Event personnalisée, avec toutes les propriétés personnalisées définies pour correspondre à l'occurrence actuelle. Dans la méthode `toString()` redéfinie, la méthode d'utilitaire `formatToString()` (héritée de Event) sert à fournir une chaîne comportant le nom du type personnalisé, ainsi que les noms et valeurs de toutes ses propriétés.

Chapitre 9 : Utilisation de domaines d'application

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le rôle de la classe ApplicationDomain est de stocker un tableau des définitions ActionScript 3.0. L'ensemble du code d'un fichier SWF est défini de sorte à exister dans un domaine d'application. Les domaines d'application servent à partitionner les classes qui se trouvent dans un même domaine de sécurité. Ainsi, plusieurs définitions de la même classe peuvent exister et les enfants peuvent réutiliser les définitions des parents.

Vous pouvez faire appel aux domaines d'application lors du chargement, au moyen de l'API de la classe Loader, d'un fichier SWF externe écrit en ActionScript 3.0 (Notez que vous ne pouvez pas utiliser les domaines d'application lorsque vous chargez une image ou un fichier SWF écrit en ActionScript 1.0 ou 2.0.) Toutes les définitions ActionScript 3.0 contenues dans la classe chargée sont stockées dans le domaine d'application. Lorsque vous chargez un fichier SWF, vous devez indiquer que le fichier doit être inclus dans le même domaine d'application que l'objet Loader en attribuant au paramètre `applicationDomain` de l'objet LoaderContext la valeur `ApplicationDomain.currentDomain`. Si vous placez le fichier SWF chargé dans le même domaine d'application, vous pourrez accéder directement à ses classes, Cela s'avère pratique si vous chargez un fichier SWF qui contient des médias incorporés auxquels vous pouvez accéder via les noms de classe associés, ou si vous voulez accéder aux méthodes du fichier SWF chargé.

L'exemple suivant présuppose l'accès à un fichier Greeter.swf distinct définissant une méthode publique nommée welcome() :

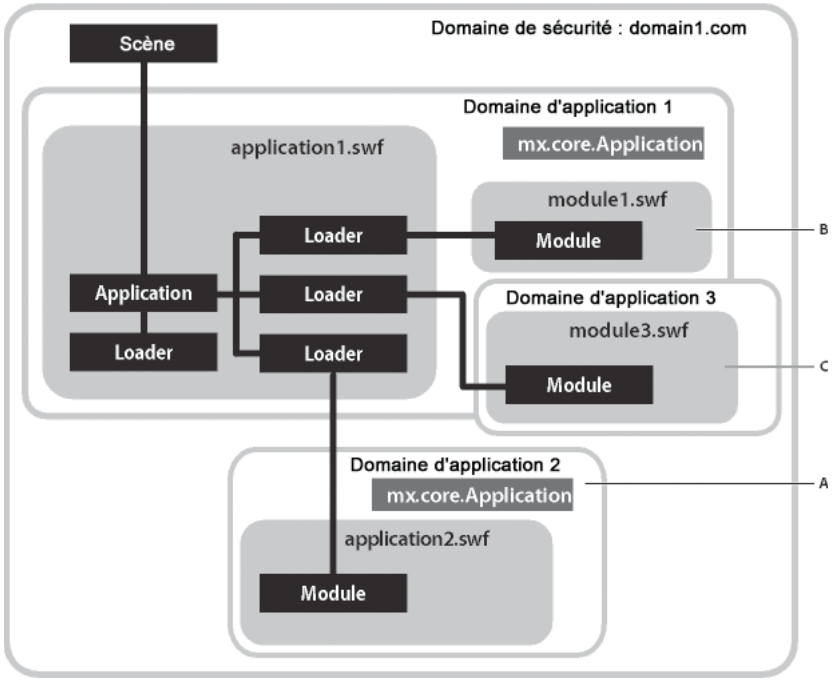
```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLRequest;
    import flash.system.ApplicationDomain;
    import flash.system.LoaderContext;

    public class ApplicationDomainExample extends Sprite
    {
        private var ldr:Loader;
        public function ApplicationDomainExample()
        {
            ldr = new Loader();
            var req:URLRequest = new URLRequest("Greeter.swf");
            var ldrContext:LoaderContext = new LoaderContext(false,
ApplicationDomain.currentDomain);
            ldr.contentLoaderInfo.addEventListener(Event.COMPLETE, completeHandler);
            ldr.load(req, ldrContext);
        }
        private function completeHandler(event:Event):void
        {
            var myGreeter:Class = ApplicationDomain.currentDomain.getDefinition("Greeter") as
Class;
            var myGreeter:Greeter = Greeter(event.target.content);
            var message:String = myGreeter.welcome("Tommy");
            trace(message); // Hello, Tommy
        }
    }
}
```

Voir aussi l'exemple de classe ApplicationDomain dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Voici d'autres points à garder à l'esprit lorsque vous utilisez les domaines d'application :

- L'ensemble du code d'un fichier SWF est défini de sorte à exister dans un domaine d'application. L'application principale s'exécute dans le *domaine d'application actif*. Le *domaine du système* contient tous les domaines d'application, y compris le domaine actif ; il contient donc toutes les classes Flash Player.
- A l'exception du domaine du système, tous les domaines d'application sont associés à un domaine parent. Le domaine parent du domaine de l'application principale est le domaine du système. Les classes chargées ne sont définies que si leur parent ne les définit pas encore. Vous ne pouvez pas remplacer une définition de classe chargée par une définition plus récente.

Le schéma suivant illustre une application qui charge du contenu à partir de divers fichiers SWF au sein d'un domaine unique, domain1.com. Selon le contenu chargé, différents domaines d'application peuvent être utilisés. Le texte suivant décrit la logique utilisée pour définir le domaine d'application approprié pour chaque fichier SWF de l'application.



***A.** Utilisation A **B.** Utilisation B **C.** Utilisation C*

Le fichier principal d'application est application1.swf. Il contient des objets Loader qui chargent du contenu à partir d'autres fichiers SWF. Dans ce scénario, le domaine d'application 1 est actif. Utilisation A, Utilisation B et Utilisation C illustrent les différentes techniques permettant de définir le domaine d'application approprié pour chaque fichier SWF de l'application.

Utilisation A Partitionnez le fichier SWF enfant en créant un enfant du domaine du système. Dans le schéma, le domaine d'application 2 est créé en tant qu'enfant du domaine du système. Le fichier application2.swf est chargé dans le domaine d'application 2 et ses définitions de classe sont ainsi partitionnées à partir des classes définies dans application1.swf.

Cette technique s'appliquera par exemple lorsqu'une ancienne application doit charger dynamiquement une nouvelle version de la même application sans créer de conflits. Les conflits sont éliminés parce que même si les noms de classe sont les mêmes, ils sont répartis dans différents domaines d'application.

Le code suivant crée un domaine d'application qui est un enfant du domaine du système, puis commence à charger un fichier SWF à l'aide de ce domaine d'application :

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

Utilisation B Ajoutez de nouvelles définitions de classe aux définitions actuelles. Le domaine d'application de module1.swf est défini sur le domaine actif (Domaine d'application 1). Vous pouvez alors ajouter au jeu actuel de définitions de classe de l'application de nouvelles définitions de classe. Cela pourrait servir pour une bibliothèque d'exécution partagée appartenant à l'application principale. Le fichier SWF est traité comme une bibliothèque partagée distante (RSL, remote shared library). Utilisez cette technique pour charger des RSL à l'aide d'un fichier de préchargement avant le lancement de l'application.

Le code suivant charge un fichier SWF, en définissant son domaine d'application sur le domaine actif :

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

Utilisation C Utilisez les définitions de classe du parent en ajoutant un nouveau domaine enfant au domaine actif. Le domaine d'application de module3.swf est un enfant du domaine actif, qui utilise pour toutes les classes les versions du parent. Cette technique peut s'appliquer à un module d'une application Internet riche (RIA, Rich Internet Application) à plusieurs écrans, qui serait chargé comme enfant de l'application principale et utiliserait les types de cette dernière. Si vous pouvez garantir que toutes les classes sont toujours mises à jour pour rester compatibles avec les anciennes versions et que l'application de chargement est toujours plus récente que les contenus qu'elle charge, les enfants utiliseront les versions des parents. L'utilisation d'un nouveau domaine d'application permet également de décharger toutes les définitions de classe en vue du nettoyage, à condition de veiller à ce qu'il ne subsiste aucune référence au fichier SWF enfant.

Cette technique autorise les modules chargés à partager les objets Singleton et les membres de classe statiques de l'objet Loader.

Le code suivant crée un domaine enfant dans le domaine actif et commence à charger un fichier SWF à l'aide de ce domaine d'application :

```
var appDomainC:ApplicationDomain = new ApplicationDomain(ApplicationDomain.currentDomain);

var contextC:LoaderContext = new LoaderContext(false, appDomainC);
var loaderC:Loader = new Loader();
loaderC.load(new URLRequest("module3.swf"), contextC);
```

Chapitre 10 : Programmation de l'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

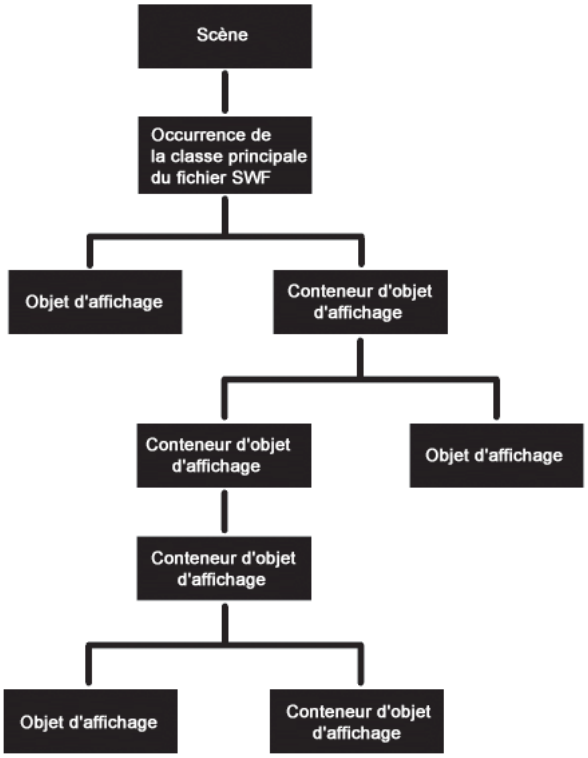
La programmation des éléments visuels dans Adobe® ActionScript® 3.0 repose sur l'utilisation des objets d'affichage sur la scène. Vous pouvez, par exemple, ajouter, déplacer, supprimer et trier les objets d'affichage, appliquer des filtres et des masques, dessiner des vecteurs et des graphiques bitmap, et exécuter des transformations en trois dimensions par le biais de l'API de programmation de l'affichage ActionScript. Les classes principales permettant de programmer l'affichage font partie du package flash.display.

Remarque : *Adobe® AIR™ fournit l'objet HTMLoader pour rendre et afficher un contenu HTML. L'objet HTMLLoader effectue le rendu des éléments visuels du DOM HTML en tant qu'objet d'affichage unique. Il est impossible d'accéder directement à chaque élément du DOM par le biais de la hiérarchie de la liste d'affichage ActionScript. Vous accédez aux éléments du DOM à l'aide de l'API DOM distincte proposée par l'objet HTMLLoader.*

Concepts fondamentaux de la programmation de l'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque application créée par le biais d'ActionScript 3.0 possède une hiérarchie d'objets d'affichage appelée *liste d'affichage*, comme l'indique l'illustration ci-dessous. La liste d'affichage contient tous les éléments visibles de l'application.



Comme le montre cette illustration, les éléments d'affichage se rangent dans un ou plusieurs groupes suivants :

- Scène

  La scène constitue le conteneur de base des objets d'affichage. Chaque application comporte un objet Stage, qui contient tous les objets d'affichage à l'écran. La scène correspond au conteneur de plus haut niveau et domine la hiérarchie de la liste d'affichage :

  Chaque fichier SWF est associé à une classe ActionScript, appelée *classe principale du fichier SWF*. Lorsqu'un fichier SWF s'ouvre dans Flash Player ou Adobe AIR, Flash Player ou AIR appelle la fonction constructeur correspondant à la classe et l'occurrence créée (systématiquement un type d'objet d'affichage) est ajoutée en tant qu'enfant de l'objet Stage. La classe principale d'un fichier SWF étend systématiquement la classe Sprite (pour plus d'informations, voir « Avantages de l'utilisation de la liste d'affichage » à la page 162).

  Vous pouvez accéder à la scène via la propriété `stage` de toute occurrence de DisplayObject. Pour plus d'informations, voir « Définition des propriétés de la scène » à la page 171.

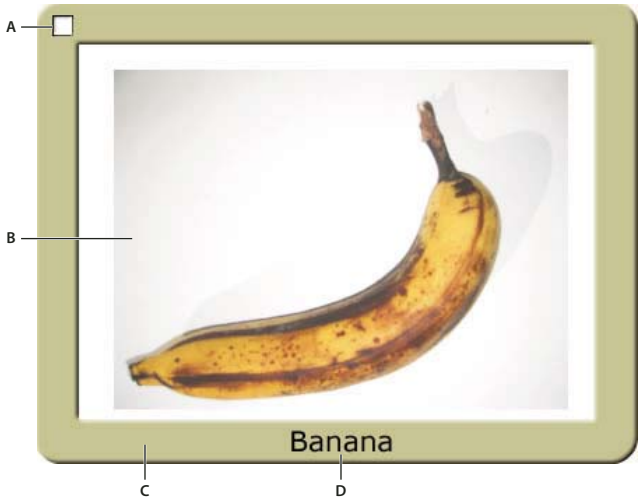
- Objets d'affichage

Dans ActionScript 3.0, tous les éléments qui apparaissent à l'écran dans une application sont des types d'*objets d'affichage*. Le package flash.display comprend une classe DisplayObject, qui correspond à une classe de base étendue par diverses autres classes. Ces autres classes représentent divers types d'objets d'affichage, tels que les formes vectorielles, les clips et les champs de texte, pour n'en citer que quelques-uns. Pour une présentation de ces classes, voir « Avantages de l'utilisation de la liste d'affichage » à la page 162.

- Conteneurs d'objets d'affichage

  Les conteneurs d'objets d'affichage sont des types spéciaux d'objets d'affichage qui, outre leur propre représentation visuelle, peuvent également comporter des objets enfant qui sont aussi des objets d'affichage.

  La classe DisplayObjectContainer est une sous-classe de la classe DisplayObject. Un objet DisplayObjectContainer peut contenir plusieurs objets d'affichage dans la *liste d'enfants* correspondante. Par exemple, l'illustration suivante contient un type d'objet DisplayObjectContainer appelé Sprite qui comporte divers objets d'affichage :



***A.** Objet SimpleButton. Ce type d'objet d'affichage possède des états « up », « down » et « over ». **B.** Objet Bitmap. Dans ce cas de figure, l'objet Bitmap a été chargé à partir d'un JPEG externe via un objet Loader. **C.** Objet Shape. Le « cadre d'image » contient un rectangle arrondi dessiné dans ActionScript. Un filtre Ombre portée est appliqué à cet objet Shape. **D.** Objet TextField.*

Dans le contexte des objets d'affichage, les objets DisplayObjectContainer portent également le nom de *conteneurs d'objets d'affichage* voire, tout simplement, de *conteneurs*. Comme indiqué précédemment, la scène est un conteneur d'objets d'affichage.

Bien que tous les objets d'affichage visibles héritent leurs caractéristiques de la classe DisplayObject, le type de chacun d'eux correspond à une sous-classe déterminée de la classe DisplayObject. Il existe, par exemple, une fonction constructeur associée à la classe Shape ou à la classe Video, mais aucune fonction constructeur pour la classe DisplayObject.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants utilisés dans le cadre de la programmation des graphiques ActionScript :

Alpha Valeur colorimétrique représentant le montant de transparence (ou, plus précisément, le montant d'opacité) d'une couleur. Ainsi, une couleur dotée d'une valeur de canal alpha de 60 % n'affiche que 60 % de son intensité totale et est transparente à 40 %.

Graphique bitmap Graphique défini en termes informatiques sous forme de grille (lignes et colonnes) de pixels de couleur. Les exemples courants de graphiques bitmap incluent les photos numériques et images similaires.

Mode de fondu Indique l'interaction requise du contenu de deux images qui se chevauchent. En règle générale, une image opaque superposée à une autre image se contente de bloquer l'image placée sous elle, qui est donc totalement invisible. Toutefois, divers modes de fondu entraînent le mélange des couleurs de diverses façons de sorte que le résultat corresponde à une combinaison des deux images.

Liste d'affichage Hiérarchie des objets d'affichage rendus sous forme de contenu visible à l'écran par Flash Player et AIR. La scène correspond à la racine de la liste d'affichage et tous les objets d'affichage associés à la scène ou à l'un de ses enfants composent la liste d'affichage (même si l'objet n'est pas à proprement parler rendu, parce qu'il réside en dehors de la scène, par exemple).

Objet d'affichage Objet représentant un type de contenu visuel dans Flash Player ou AIR. La liste d'affichage ne contient que des objets d'affichage et toutes les classes d'objets d'affichage sont des sous-classes de la classe DisplayObject.

Conteneur d'objet d'affichage Type spécial d'objet d'affichage qui, outre (généralement) sa propre représentation visuelle, peut comporter des objets d'affichage enfant.

Classe principale du fichier SWF Classe qui définit le comportement de l'objet d'affichage de plus haut niveau d'un fichier SWF, soit, fondamentalement, la classe associée au fichier SWF en tant que tel. Ainsi, dans un fichier SWF généré dans un outil de création Flash, la classe principale correspond à la classe du document. Elle possède un « scénario principal » qui intègre tous les autres scénarios. La classe principale du fichier SWF correspond à la classe dont le scénario principal est une occurrence.

Masquage Technique consistant à ne pas afficher certaines parties d'une image (ou, à l'inverse, à n'afficher que certaines parties d'une image). Les sections de l'image masque deviennent transparentes, afin d'assurer la visibilité du contenu sous-jacent. Ce terme se réfère à la bande utilisée par un peintre en bâtiment pour empêcher la peinture d'être appliquée à certaines sections.

Scène Conteneur visuel correspondant à la base ou à l'arrière-plan de tout contenu visuel dans un fichier SWF.

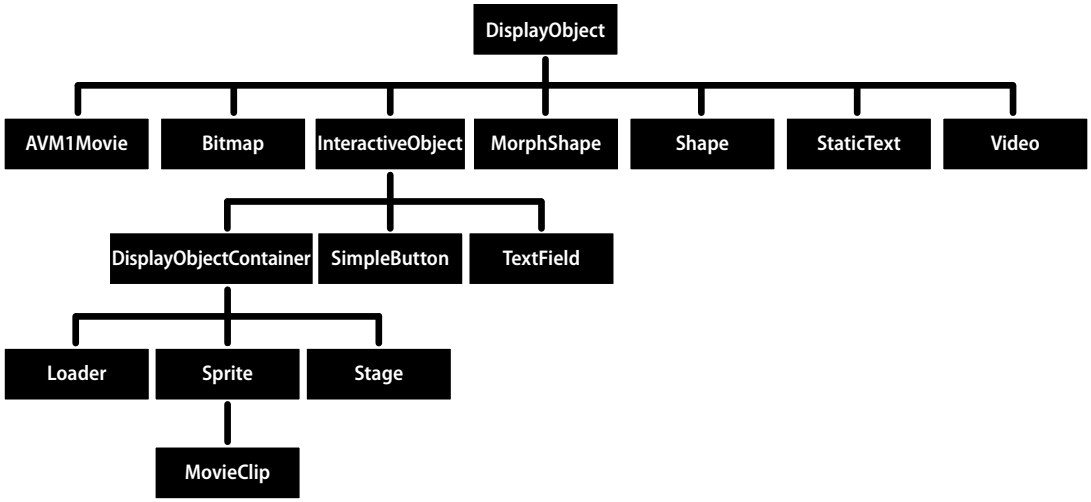
Transformation Modification des caractéristiques visuelles d'un graphique (rotation de l'objet, modification de son échelle, désalignement, déformation ou altération de sa couleur).

Graphique vectoriel Graphique défini en termes informatiques par des lignes et des formes dessinées en fonction de caractéristiques déterminées (épaisseur, longueur, taille, angle et position, par exemple).

Classes d'affichage de base

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le package flash.display ActionScript 3.0 contient des classes destinées aux objets visuels susceptibles d'apparaître dans Flash Player ou AIR. L'illustration suivante identifie les relations entre les sous-classes de ces classes d'objets d'affichage de base.



L'illustration indique ce dont héritent les classes d'objets d'affichage. Notez que certaines de ces classes, en particulier StaticText, TextField et Video, ne figurent pas dans le package flash.display, mais héritent toutefois des caractéristiques de la classe DisplayObject.

Toutes les classes qui étendent la classe DisplayObject héritent de ses méthodes et propriétés. Pour plus d'informations, voir « Propriétés et méthodes de la classe DisplayObject » à la page 165.

Vous pouvez créer une occurrence d'un objet des classes suivantes, qui figurent dans le package flash.display :

- Bitmap : la classe Bitmap permet de définir des objets bitmap, qu'ils soient chargés à partir de fichiers externes ou rendus via ActionScript. Vous pouvez charger des bitmaps à partir de fichiers externes par le biais de la classe Loader. Libre à vous de charger des fichiers GIF, JPG ou PNG. Vous pouvez également créer un objet BitmapData à partir de données personnalisées, puis créer un objet Bitmap qui utilise ces données. Les méthodes de la classe BitmapData permettent de modifier les bitmaps, qu'ils soient chargés ou créés dans ActionScript. Pour plus d'informations, voir « Chargement d'objets d'affichage » à la page 206 et le chapitre « Utilisation des images bitmap » à la page 251.
- Loader : la classe Loader permet de charger des ressources externes (fichiers SWF ou graphiques). Pour plus d'informations, voir « Chargement dynamique du contenu d'affichage » à la page 205.
- Shape : la classe Shape permet de créer des graphiques vectoriels, tels que des rectangles, des lignes, des cercles, etc. Pour plus d'informations, voir « Utilisation de l'API de dessin » à la page 230.
- SimpleButton : un objet SimpleButton est une représentation ActionScript d'un symbole de bouton créé dans l'outil de programmation Flash. Une occurrence de SimpleButton est dotée de quatre états de bouton : « up », « down », « over » et « hit test » (zone qui réagit aux événements souris et clavier).

- Sprite : un objet Sprite peut contenir des graphiques qui lui sont propres, ainsi que des objets d'affichage enfant (la classe Sprite étend la classe DisplayObjectContainer). Pour plus d'informations, voir « Utilisation de conteneurs d'objets d'affichage » à la page 165 et « Utilisation de l'API de dessin » à la page 230.
- MovieClip : un objet MovieClip est la forme ActionScript d'un symbole de clip créé dans l'outil de programmation Flash. En pratique, un objet MovieClip est similaire à un objet Sprite, à une exception près : il possède également un scénario. Pour plus d'informations, voir « Utilisation des clips » à la page 333.

Les classes suivantes, qui ne figurent pas dans le package flash.display, sont des sous-classes de la classe DisplayObject :

- La classe TextField, qui figure dans le package flash.text, est un objet d'affichage destiné à l'affichage et à la saisie de texte. Pour plus d'informations, voir « Principes de base de l'utilisation du texte » à la page 383.
- La classe TextLine, qui figure dans le package flash.text.engine, correspond à l'objet d'affichage permettant d'afficher des lignes de texte composées par Flash Text Engine et Text Layout Framework. Pour plus d'informations, voir « Utilisation de Flash Text Engine » à la page 410 et « Utilisation de Text Layout Framework » à la page 440.
- La classe Video, qui figure dans le package flash.media, correspond à l'objet d'affichage utilisé pour afficher des fichiers vidéo. Pour plus d'informations, voir « Utilisation de la vidéo » à la page 489.

Les classes suivantes du package flash.display étendent la classe DisplayObject, mais il est impossible d'en créer une occurrence. Parce qu'elles combinent des fonctionnalités communes en une classe unique, elles servent plutôt de classes parent à d'autres objets d'affichage.

- AVM1Movie : la classe AVM1Movie permet de représenter des fichiers SWF chargés créés dans ActionScript 1.0 et 2.0.
- DisplayObjectContainer : les classes Loader, Stage, Sprite et MovieClip étendent chacune la classe DisplayObjectContainer. Pour plus d'informations, voir « Utilisation de conteneurs d'objets d'affichage » à la page 165.
- InteractiveObject : classe de base de tous les objets utilisés pour interagir avec la souris et le clavier. Les objets SimpleButton, TextField, Loader, Sprite, Stage et MovieClip sont tous des sous-classes de la classe InteractiveObject. Pour plus d'informations sur la création d'une interaction de souris ou de clavier, voir « Principes de base de l'interaction utilisateur » à la page 574.
- MorphShape : ces objets sont générés lors de la création d'une interpolation de forme dans l'outil de programmation Flash. Il est impossible d'en créer des occurrences par le biais d'ActionScript, mais vous pouvez y accéder dans la liste d'affichage.
- Scène : la classe Stage étend la classe DisplayObjectContainer. Il n'existe qu'une seule occurrence de scène par application, et elle figure au sommet de la hiérarchie de la liste d'affichage. Vous pouvez accéder à la scène via la propriété `stage` de toute occurrence de DisplayObject. Pour plus d'informations, voir « Définition des propriétés de la scène » à la page 171.

Par ailleurs, la classe StaticText, qui figure dans le package flash.text, étend la classe DisplayObject, mais il est impossible d'en créer une occurrence dans du code. Les champs de texte statique sont créés dans Flash uniquement.

Les classes suivantes ne sont ni des objets d'affichage, ni des conteneurs d'objets d'affichage et n'apparaissent pas dans la liste d'affichage, mais affichent des graphiques sur la scène. Elles dessinent des éléments dans un rectangle, appelé fenêtre d'affichage, positionné relativement à la scène.

- StageVideo : la classe StageVideo affiche le contenu vidéo en faisant appel, dans la mesure du possible, à l'accélération matérielle. Cette classe est disponible à partir de Flash Player 10.2. Pour plus d'informations, voir « Présentation à accélération matérielle par le biais de la classe StageVideo » à la page 528.
- StageWebView : la classe StageWebView affiche le contenu HTML. Elle est prise en charge depuis la version 2.5 d'AIR. Pour plus d'informations, voir « Objets StageWebView » à la page 1068.

Les classes fl.display suivantes proposent des fonctionnalités équivalentes à celles des classes flash.display.Loader et LoaderInfo. Utilisez-les au lieu des classes flash.display équivalentes en cas de développement dans l'environnement Flash Professional (CS5.5 ou ultérieur). Dans cet environnement, ces classes permettent de résoudre les problèmes liés à TLF avec préchargement RSL. Pour plus d'informations, voir « Utilisation des classes ProLoader et ProLoaderInfo » à la page 210.

- fl.display.ProLoader : équivalente à flash.display.Loader
- fl.display.ProLoaderInfo : équivalente à flash.display.LoaderInfo

Avantages de l'utilisation de la liste d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 3.0, des classes distinctes sont réservées aux différents types d'objets d'affichage. Dans ActionScript 1.0 et 2.0, un grand nombre de types d'objets identiques sont inclus dans une même classe : MovieClip.

Cette individualisation des classes et la structure hiérarchique des listes d'affichage présentent les avantages suivants :

- Rendu plus efficace et utilisation réduite de la mémoire
- Gestion optimisée de la profondeur
- Parcours entier de la liste d'affichage
- Objets d'affichage absents de la liste
- Classement simplifié en sous-classes des objets d'affichage

Rendu plus efficace et taille réduite des fichiers

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 1.0 et 2.0, vous ne pouvez dessiner des formes que dans un objet MovieClip. ActionScript 3.0 intègre des classes d'objets d'affichage plus simples, dans lesquelles vous pouvez dessiner des formes. Parce que ces classes d'objets d'affichage ActionScript 3.0 ne contiennent pas le jeu complet de méthodes et propriétés associées à un objet MovieClip, elles mobilisent moins de ressources en mémoire et processeur.

Par exemple, à l'encontre d'un objet Shape, chaque objet MovieClip comporte des propriétés associées au scénario du clip. Les propriétés de gestion du scénario font parfois appel à un volume considérable de ressources en mémoire et processeur. Dans ActionScript 3.0, l'utilisation de l'objet Shape se traduit par une amélioration des performances. L'objet Shape nécessite moins de ressources que l'objet MovieClip, plus complexe. Flash Player et AIR n'ont pas besoin de gérer les propriétés MovieClip inutilisées, optimisant ainsi la vitesse et réduisant les besoins de mémoire de l'objet.

Gestion optimisée de la profondeur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 1.0 et 2.0, la profondeur était gérée par un système et des méthodes de gestion linéaire de la profondeur, tels que `getNextHighestDepth()`.

ActionScript 3.0 comprend la classe DisplayObjectContainer, dont les méthodes et propriétés sont mieux adaptées à la gestion de la profondeur des objets d'affichage.

Dans ActionScript 3.0, lorsque vous déplacez un objet d'affichage au sein de la liste des enfants d'une occurrence de DisplayObjectContainer, les autres enfants du conteneur d'objets d'affichage sont automatiquement repositionnés et des positions d'index enfant appropriées leur sont affectées dans le conteneur d'objets d'affichage.

Par ailleurs, ActionScript 3.0 permet systématiquement de détecter tous les objets enfant de tout conteneur d'objets d'affichage. Chaque occurrence de DisplayObjectContainer possède une propriété `numChildren`, qui indique le nombre d'enfants figurant dans le conteneur d'objets d'affichage. Puisque la liste des enfants d'un conteneur d'objets d'affichage correspond systématiquement à une liste indexée, vous pouvez examiner chaque objet de la liste de la position d'index 0 à la dernière position d'index (`numChildren - 1`). Cette technique n'était pas proposée par les méthodes et propriétés d'un objet MovieClip dans ActionScript 1.0 et 2.0.

ActionScript 3.0 permet de parcourir aisément et séquentiellement la liste d'affichage, car les numéros d'index de la liste des enfants d'un conteneur d'objets d'affichage se suivent. Parcourir la liste d'affichage et gérer la profondeur des objets est désormais beaucoup plus simple que dans ActionScript 1.0 et 2.0. Dans ActionScript 1.0 et 2.0, un clip pouvait en effet contenir des objets dont l'ordre de profondeur n'était pas séquentiel, ce qui rendait parfois le parcours de la liste d'objets difficile. Dans ActionScript 3.0, chaque liste d'enfants d'un conteneur d'objets d'affichage est mise en cache en interne sous forme de tableau, ce qui permet des recherches extrêmement rapides (par index). Passer en boucle sur tous les enfants d'un conteneur d'objets d'affichage s'avère également très rapide.

Dans ActionScript 3.0, vous pouvez également accéder aux enfants d'un conteneur d'objets d'affichage par le biais de la méthode `getChildByName()` de la classe DisplayObjectContainer.

Parcours entier de la liste d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 1.0 et 2.0 ne vous permettaient pas d'accéder à certains objets, telles les formes vectorielles, dessinées dans l'outil de programmation Flash. Dans ActionScript 3.0, vous pouvez accéder à tous les objets de la liste d'affichage, qu'ils aient été créés en ActionScript ou dans l'outil de programmation Flash. Pour plus d'informations, voir « Parcours de la liste d'affichage » à la page 169.

Objets d'affichage absents de la liste

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 permet de créer des objets d'affichage qui ne figurent pas dans la liste d'affichage visible. Ils portent le nom d'objets d'affichage *hors liste*. Un objet d'affichage n'est ajouté à la liste d'affichage visible que lorsque vous appelez la méthode `addChild()` ou `addChildAt()` d'une occurrence de DisplayObjectContainer qui a déjà été intégrée à la liste d'affichage.

Les objets d'affichage hors liste permettent d'assembler des objets d'affichage complexes, tels que ceux qui possèdent plusieurs conteneurs d'objets d'affichage comportant plusieurs objets d'affichage. En n'intégrant pas à la liste des objets d'affichage, vous pouvez assembler des objets complexes sans avoir à effectuer leur rendu. Vous économisez ainsi le temps de traitement correspondant. Vous pouvez alors ajouter un objet hors liste à la liste d'affichage au moment voulu. Il est également possible d'intégrer un enfant d'un conteneur d'objets d'affichage à la liste d'affichage, puis de l'en extraire ou d'en modifier la position dans cette dernière, le cas échéant.

Classement simplifié en sous-classes des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 1.0 et 2.0, il était souvent nécessaire d'ajouter de nouveaux objets MovieClip à un fichier SWF pour créer des formes de base ou afficher des bitmaps. Dans ActionScript 3.0, la classe DisplayObject comprend un grand nombre de sous-classes intégrées, telles que Shape et Bitmap. Parce que les classes d'ActionScript 3.0 sont plus spécialisées pour des types spécifiques d'objets, il est plus simple de créer des sous-classes de base des classes intégrées.

Par exemple, pour dessiner un cercle dans ActionScript 2.0, vous pourriez créer une classe CustomCircle qui étend la classe MovieClip lors de la création d'une occurrence d'un objet de la classe personnalisée. Néanmoins, cette classe comprendrait également diverses propriétés et méthodes émanant de la classe MovieClip (telles que `totalFrames`) qui ne s'appliquent pas à elle. Dans ActionScript 3.0, vous pouvez toutefois créer une classe CustomCircle qui étend l'objet Shape et, de ce fait, ne comprend pas les propriétés et méthodes sans rapport contenues dans la classe MovieClip. Le code suivant illustre un exemple de classe CustomCircle :

```
import flash.display.*;

public class CustomCircle extends Shape
{
    var xPos:Number;
    var yPos:Number;
    var radius:Number;
    var color:uint;
    public function CustomCircle(xInput:Number,
                                 yInput:Number,
                                 rInput:Number,
                                 colorInput:uint)
    {
        xPos = xInput;
        yPos = yInput;
        radius = rInput;
        color = colorInput;
        this.graphics.beginFill(color);
        this.graphics.drawCircle(xPos, yPos, radius);
    }
}
```

Utilisation des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Maintenant que vous maîtrisez les bases de la scène, des objets d'affichage, des conteneurs d'objets d'affichage et de la liste d'affichage, cette section contient des informations plus détaillées relatives à l'utilisation des objets d'affichage dans ActionScript 3.0.

Propriétés et méthodes de la classe DisplayObject

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tous les objets d'affichage sont des sous-classes de la classe DisplayObject et, de ce fait, héritent des propriétés et méthodes de cette dernière. Les propriétés dont ils héritent correspondent aux propriétés de base qui s'appliquent à tous les objets d'affichage. Par exemple, chaque objet d'affichage possède une propriété `x` et une propriété `y` qui indiquent sa position dans son conteneur d'objets d'affichage.

Il est impossible de créer une occurrence de DisplayObject à l'aide du constructeur de la classe DisplayObject. Vous devez créer un autre type d'objet (un objet qui est une sous-classe de la classe DisplayObject), tel Sprite, pour créer une occurrence d'objet par le biais de l'opérateur `new`. Par ailleurs, pour créer une classe d'objet d'affichage personnalisée, vous devez créer une sous-classe de l'une des sous-classes d'objets d'affichage ayant une fonction constructeur utilisable (telle que la classe Shape ou la classe Sprite). Pour plus d'informations, voir la description de la classe DisplayObject dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Ajout d'objets d'affichage à la liste d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez une occurrence d'un objet d'affichage, elle n'apparaît pas à l'écran (sur la scène) tant que vous ne l'avez pas ajoutée à un conteneur d'objets d'affichage figurant dans la liste d'affichage. Par exemple, dans le code suivant, l'objet `myText` TextField n'est pas visible si vous omettez la dernière ligne de code. Dans la dernière ligne de code, le mot-clé `this` doit se référer à un conteneur d'objets d'affichage figurant déjà dans la liste d'affichage.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

Lorsque vous ajoutez un élément visuel à la scène, il devient un *enfant* de cette dernière. Le premier fichier SWF chargé dans une application (tel celui intégré à une page HTML) est automatiquement ajouté en tant qu'enfant de la scène. Il peut s'agir de n'importe quel type d'objet qui étend la classe Sprite.

Tout objet d'affichage créé *sans* utiliser ActionScript (par exemple en ajoutant une balise MXML dans un fichier MXML Flex ou en plaçant un élément sur la scène dans Flash Professional) est ajouté à la liste d'affichage. Bien que vous n'ajoutiez pas ces objets d'affichage par le biais d'ActionScript, vous pouvez y accéder via ActionScript. Par exemple, le code suivant règle la largeur d'un objet appelé `button1`, qui a été ajouté dans l'outil de programmation (et non via ActionScript) :

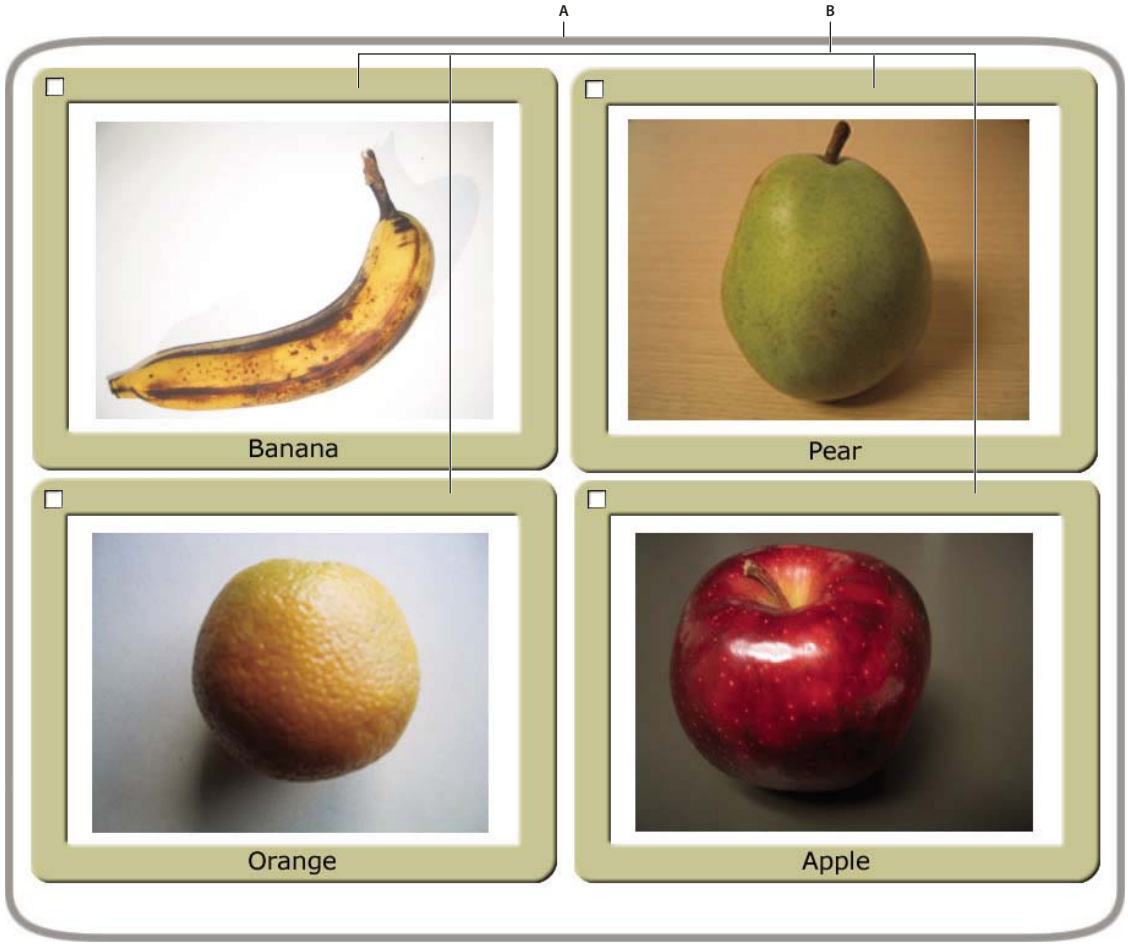
```
button1.width = 200;
```

Utilisation de conteneurs d'objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si un objet DisplayObjectContainer est supprimé de la liste d'affichage ou s'il est transféré ou transformé d'une autre façon, chaque objet d'affichage de DisplayObjectContainer est également supprimé, transféré ou transformé.

Un conteneur d'objets d'affichage correspond à un type d'objet d'affichage et peut être ajouté à un autre conteneur d'objets d'affichage. Par exemple, l'image suivante illustre un conteneur d'objets d'affichage, `pictureScreen`, qui comporte une forme de contour et quatre autres conteneurs d'objets d'affichage (de type PictureFrame) :



A. *Forme définissant la bordure du conteneur d'objets d'affichage pictureScreen* ***B.*** *Quatre conteneurs d'objets d'affichage, qui sont des enfants de l'objet pictureScreen*

Pour qu'un objet d'affichage apparaisse dans la liste d'affichage, vous devez l'ajouter à un conteneur d'objets d'affichage figurant dans la liste d'affichage. A cet effet, vous utilisez la méthode `addChild()` ou la méthode `addChildAt()` de l'objet conteneur. Par exemple, sans la dernière ligne du code suivant, l'objet `myTextField` ne s'afficherait pas :

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

Dans cet exemple de code, `this.root` pointe vers le conteneur d'objets d'affichage MovieClip qui comporte le code. Dans votre propre code, vous pouvez stipuler un autre conteneur.

Utilisez la méthode `addChildAt()` pour ajouter l'enfant à une position déterminée de la liste des enfants du conteneur d'objets d'affichage. Ces positions d'index basées sur zéro dans la liste des enfants se réfèrent à l'ordre d'apparition (de l'avant à l'arrière) des objets d'affichage. Considérons par exemple les trois objets d'affichage suivants. Chaque objet a été créé à partir d'une classe personnalisée appelée Ball.

L'ordre d'apparition de ces objets d'affichage dans leur conteneur peut être modifié par le biais de la méthode `addChildAt()`. Considérons par exemple le code qui suit :

```
ball_A = new Ball(0xFFCC00, "a");
ball_A.name = "ball_A";
ball_A.x = 20;
ball_A.y = 20;
container.addChild(ball_A);

ball_B = new Ball(0xFFCC00, "b");
ball_B.name = "ball_B";
ball_B.x = 70;
ball_B.y = 20;
container.addChild(ball_B);

ball_C = new Ball(0xFFCC00, "c");
ball_C.name = "ball_C";
ball_C.x = 40;
ball_C.y = 60;
container.addChildAt(ball_C, 1);
```

Une fois ce code exécuté, les objets d'affichage sont placés comme suit dans l'objet DisplayObjectContainer `container`. Notez l'ordre d'apparition des objets.

Pour placer un objet en tête de la liste d'affichage, il suffit de l'ajouter à nouveau à celle-ci. Par exemple, après le code précédent, utilisez la ligne de code suivante pour placer `ball_A` en première position dans la pile :

```
container.addChild(ball_A);
```

Ce code supprime `ball_A` de son emplacement actuel dans la liste d'affichage de `container`, et l'ajoute ensuite au sommet de la liste, ce qui a pour effet de le placer en haut de l'empilement d'objets.

Vous disposez de la méthode `getChildAt()` pour vérifier l'ordre d'apparition des objets d'affichage. La méthode `getChildAt()` renvoie les objets enfant d'un conteneur en fonction du numéro d'index transmis. Par exemple, le code suivant révèle le nom des objets d'affichage placés à des positions diverses dans la liste des enfants de l'objet DisplayObjectContainer `container` :

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

Si vous supprimez un objet d'affichage de la liste des enfants de son conteneur parent, les éléments de la liste ayant un indice plus élevé descendent tous d'une position dans l'index des enfants. Ainsi, si nous reprenons l'exemple précédent, le code ci-après illustre le transfert de l'objet d'affichage qui occupait la position 2 dans l'objet DisplayObjectContainer `container` vers la position 1 suite à la suppression d'un objet d'affichage occupant une position inférieure dans la liste d'enfants :

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

Les méthodes `removeChild()` et `removeChildAt()` ne suppriment pas entièrement une occurrence d'objet d'affichage. Elles se contentent de la supprimer de la liste des enfants du conteneur. Une autre variable peut continuer à faire référence à l'occurrence (utilisez l'opérateur `delete` pour supprimer totalement un objet).

Un objet d'affichage ne possédant qu'un seul conteneur parent, vous ne pouvez ajouter une occurrence d'objet d'affichage qu'à un seul conteneur d'objets d'affichage. Par exemple, le code suivant indique que l'objet d'affichage `tf1` ne peut figurer que dans un seul conteneur (soit, dans ce cas, un Sprite, qui étend la classe DisplayObjectContainer) :

```
tf1:TextField = new TextField();
tf2:TextField = new TextField();
tf1.name = "text 1";
tf2.name = "text 2";

container1:Sprite = new Sprite();
container2:Sprite = new Sprite();

container1.addChild(tf1);
container1.addChild(tf2);
container2.addChild(tf1);

trace(container1.numChildren); // 1
trace(container1.getChildAt(0).name); // text 2
trace(container2.numChildren); // 1
trace(container2.getChildAt(0).name); // text 1
```

Si vous ajoutez à un conteneur d'objets d'affichage un objet qui est déjà contenu dans un autre conteneur d'objets d'affichage, l'objet sera supprimé de la liste des enfants de ce dernier.

Outre les méthodes décrites précédemment, la classe DisplayObjectContainer définit plusieurs méthodes d'utilisation des objets d'affichage enfant, notamment :

- `contains()` : détermine si un objet d'affichage est un enfant d'un objet DisplayObjectContainer.
- `getChildByName()` : extrait un objet d'affichage en fonction de son nom.
- `getChildIndex()` : renvoie la position d'index d'un objet d'affichage.
- `setChildIndex()` : modifie la position d'un objet d'affichage enfant.
- `removeChildren()` : supprime plusieurs objets d'affichage enfants.

- `swapChildren()` : permute l'ordre de deux objets d'affichage.
- `swapChildrenAt()` : permute l'ordre de deux objets d'affichage définis en fonction de leur valeur d'index.

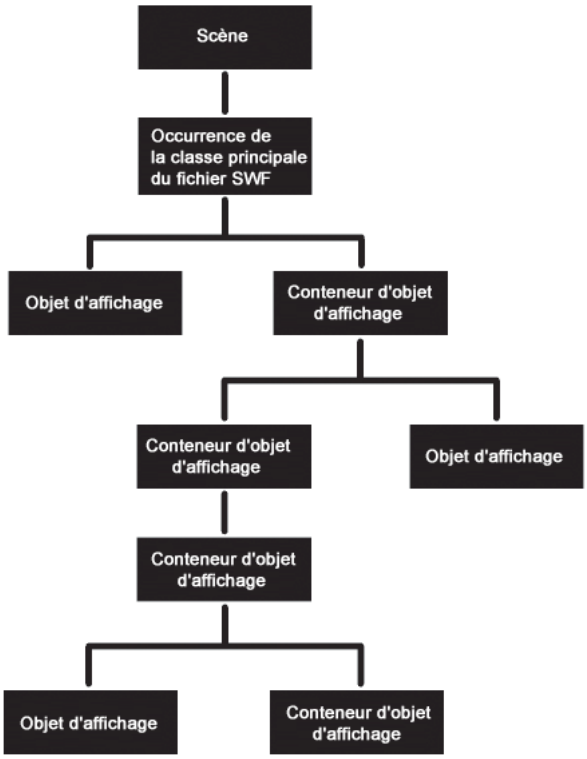
Pour plus d'informations, voir les entrées pertinentes du manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

N'oubliez pas qu'un objet d'affichage qui ne figure pas dans la liste d'affichage (donc, qui ne se trouve pas dans un conteneur d'objets d'affichage enfant de la scène) est appelé objet d'affichage *hors liste*.

Parcours de la liste d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme nous l'avons vu, la liste d'affichage est une structure en arborescence. Au sommet de l'arborescence figure la scène, qui peut comporter plusieurs objets d'affichage. Les objets d'affichage qui sont eux-mêmes des conteneurs d'objets d'affichage peuvent contenir d'autres objets d'affichage, voire des conteneurs d'objets d'affichage.



La classe DisplayObjectContainer comporte des propriétés et méthodes de parcours de la liste d'affichage, par le biais des listes d'enfants des conteneurs d'objets d'affichage. Considérons par exemple le code suivant, qui ajoute deux objets d'affichage, `title` et `pict`, à l'objet `container` (qui est un Sprite, et la classe Sprite étend la classe DisplayObjectContainer) :

```
var container:Sprite = new Sprite();
var title:TextField = new TextField();
title.text = "Hello";
var pict:Loader = new Loader();
var url:URLRequest = new URLRequest("banana.jpg");
pict.load(url);
pict.name = "banana loader";
container.addChild(title);
container.addChild(pict);
```

La méthode `getChildAt()` renvoie l'enfant de la liste d'affichage à une position d'index déterminée :

```
trace(container.getChildAt(0) is TextField); // true
```

Vous pouvez également accéder aux objets enfant en indiquant leur nom. Chaque objet d'affichage possède un nom, qui est attribué par défaut par Flash Player ou AIR (tel « `instance1` ») si vous ne l'attribuez pas vous-même. Par exemple, le code suivant indique comment utiliser la méthode `getChildByName()` pour accéder à un objet d'affichage enfant portant le nom « `banana loader` » :

```
trace(container.getChildByName("banana loader") is Loader); // true
```

L'utilisation de la méthode `getChildByName()` entraîne parfois un ralentissement des performances par rapport à la méthode `getChildAt()`.

Puisque la liste d'affichage d'un conteneur d'objets d'affichage peut contenir d'autres conteneurs d'objets d'affichage en tant qu'objets enfant, vous pouvez parcourir la liste d'affichage complète de l'application sous forme d'arborescence. Par exemple, dans l'extrait de code illustré précédemment, une fois l'opération de chargement de l'objet Loader `pict` terminée, un objet d'affichage enfant (l'image bitmap) de l'objet `pict` est chargé. Pour accéder à cet objet d'affichage bitmap, vous pouvez écrire `pict.getChildAt(0)`. Vous pouvez également écrire `container.getChildAt(0).getChildAt(0)` (puisque `container.getChildAt(0) == pict`).

La fonction suivante génère un extrait `trace()` en retrait de la liste d'affichage d'un conteneur d'objets d'affichage :

```
function traceDisplayList(container:DisplayObjectContainer,indentString:String = ""):void
{
    var child:DisplayObject;
    for (var i:uint=0; i < container.numChildren; i++)
    {
        child = container.getChildAt(i);
        trace(indentString, child, child.name);
        if (container.getChildAt(i) is DisplayObjectContainer)
        {
            traceDisplayList(DisplayObjectContainer(child), indentString + "")
        }
    }
}
```

Adobe Flex

Si vous utilisez Flex, notez que cette application définit un grand nombre de classes d'objets d'affichage de composant et que celles-ci priment sur les méthodes d'accès à la liste d'affichage de la classe DisplayObjectContainer. Par exemple, la classe Container du package mx.core prime sur la méthode `addChild()` et autres méthodes de la classe DisplayObjectContainer (étendue par la classe Container). Dans le cas de la méthode `addChild()`, la classe primant sur la méthode, vous ne pouvez pas ajouter tous les types d'objets d'affichage à une occurrence de Container dans Flex. La méthode remplacée demande dans ce cas que l'objet enfant ajouté soit de type mx.core.UIComponent.

Définition des propriétés de la scène

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Stage annule la plupart des propriétés et méthodes de la classe DisplayObject. Si vous appelez l'une de ces propriétés ou méthodes annulées, Flash Player et AIR renvoient une exception. Par exemple, l'objet Stage ne possède pas de propriété `x` ou `y` car, en tant que conteneur principal de l'application, sa position est fixe. Or, les propriétés `x` et `y` indiquent la position d'un objet d'affichage par rapport à son conteneur, et puisque la scène ne se trouve pas dans un autre conteneur d'objets d'affichage, ces propriétés seraient inapplicables.

Remarque : *certaines propriétés et méthodes de la classe Stage sont réservées aux objets d'affichage appartenant au même sandbox de sécurité que le premier fichier SWF chargé. Pour plus d'informations, voir* « *Sécurité de la scène* » *à la page 1109.*

Contrôle de la cadence de lecture

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `framerate` de la classe Stage permet de définir la cadence de tous les fichiers SWF chargés dans l'application. Pour plus d'informations, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Contrôle de la mise à l'échelle de la scène

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque la portion de l'écran qui représente Flash Player ou AIR est redimensionnée, le moteur d'exécution ajuste automatiquement le contenu de la scène en conséquence. La propriété `scaleMode` de la classe Stage détermine comment le contenu de la scène est ajusté. Cette propriété peut être réglée sur quatre valeurs distinctes, définies en tant que constantes dans la classe flash.display.StageScaleMode :

- `StageScaleMode.EXACT_FIT` met à l'échelle le fichier SWF de sorte à occuper les nouvelles dimensions de la scène sans tenir compte du format d'origine du contenu. Etant donné que les facteurs d'échelle ne sont pas nécessairement identiques en largeur et en hauteur, le contenu risque de sembler écrasé ou étiré en cas de modification du format de la scène.
- `StageScaleMode.SHOW_ALL` met à l'échelle le fichier SWF de sorte à occuper les nouvelles dimensions de la scène sans modifier le format du contenu. Ce mode de mise à l'échelle affiche la totalité du contenu, mais risque de donner lieu à des bordures de type « boîte aux lettres » similaires aux barres noires qui entourent un film grand écran sur une télévision standard.
- `StageScaleMode.NO_BORDER` met à l'échelle le fichier SWF de sorte à occuper totalement les nouvelles dimensions de la scène sans modifier le format du contenu. Ce mode de mise à l'échelle exploite pleinement la zone d'affichage de la scène, mais risque d'engendrer un recadrage.
- `StageScaleMode.NO_SCALE` : ne met pas à l'échelle le fichier SWF. Si les nouvelles dimensions de la scène sont inférieures aux dimensions d'origine, le contenu est recadré. Si elles leur sont supérieures, l'espace ajouté est vide.

  Dans le mode de mise à l'échelle `StageScaleMode.NO_SCALE` uniquement, les propriétés `stageWidth` et `stageHeight` de la classe Stage permettent de déterminer les dimensions réelles de la scène redimensionnée, exprimées en pixels. (Dans les autres modes, les propriétés `stageWidth` et `stageHeight` renvoient toujours la largeur et la hauteur d'origine du fichier SWF.) De plus, lorsque la propriété `scaleMode` est définie sur `StageScaleMode.NO_SCALE` et que le fichier SWF est redimensionné, l'événement `resize` de la classe Stage est distribué, ce qui vous permet d'effectuer des ajustements en conséquence.

Définir `scaleMode` sur `StageScaleMode.NO_SCALE` permet donc de mieux contrôler l'ajustement du contenu en cas de redimensionnement de la fenêtre. Par exemple, si un fichier SWF contient une vidéo et une barre de contrôle, il peut s'avérer utile de conserver la taille de la barre de contrôle en cas de redimensionnement de la scène et de ne modifier que la taille de la fenêtre de vidéo en fonction du changement de taille de la scène. Ce cas de figure est illustré dans l'exemple suivant :

```
// mainContent is a display object containing the main content;
// it is positioned at the top-left corner of the Stage, and
// it should resize when the SWF resizes.

// controlBar is a display object (e.g. a Sprite) containing several
// buttons; it should stay positioned at the bottom-left corner of the
// Stage (below mainContent) and it should not resize when the SWF
// resizes.

import flash.display.Stage;
import flash.display.StageAlign;
import flash.display.StageScaleMode;
import flash.events.Event;

var swfStage:Stage = mainContent.stage;
swfStage.scaleMode = StageScaleMode.NO_SCALE;
swfStage.align = StageAlign.TOP_LEFT;
swfStage.addEventListener(Event.RESIZE, resizeDisplay);

function resizeDisplay(event:Event):void
{
    var swfWidth:int = swfStage.stageWidth;
    var swfHeight:int = swfStage.stageHeight;

    // Resize the main content area
    var newContentHeight:Number = swfHeight - controlBar.height;
    mainContent.height = newContentHeight;
    mainContent.scaleX = mainContent.scaleY;

    // Reposition the control bar.
    controlBar.y = newContentHeight;
}
```

Définition du mode de mise à l'échelle de la scène pour les fenêtres AIR

La propriété `scaleMode` de la scène indique comment celle-ci met à l'échelle et écrête les objets d'affichage enfant lors d'un redimensionnement de fenêtre. N'utilisez que le mode `noScale` dans AIR. Lorsque ce mode est activé, la scène n'est pas mise à l'échelle. La taille de la scène est modifiée directement en fonction des limites de la fenêtre. Les objets risquent d'être écrêtés si la taille de la fenêtre est inférieure à sa taille initiale.

Les modes de mise à l'échelle de la scène sont adaptés aux environnements tels qu'un navigateur Web qui ne permet pas nécessairement de contrôler la taille ou le format de la scène. Grâce aux modes, vous pouvez sélectionner le cas de figure le moins inadapté lorsque la scène ne correspond pas à la taille ou au format idéal défini par l'application. Dans AIR, vous contrôlez toujours la scène. Par conséquent, modifier la disposition du contenu ou redimensionner la fenêtre assure un meilleur résultat que l'activation de la mise à l'échelle de la scène.

Dans le navigateur et la fenêtre AIR initiale, la relation entre la taille de fenêtre et le facteur d'échelle initial est extraite du fichier SWF chargé. Toutefois, lorsque vous créez un objet NativeWindow, AIR choisit une relation arbitraire définie sur 72:1 entre la taille de fenêtre et le facteur d'échelle. Par conséquent, si la fenêtre mesure 72x72 pixels, un rectangle de 10x10 pixels ajouté à la fenêtre est dessiné à la taille correcte (soit 10x10 pixels). Par contre, si la fenêtre mesure 144x144 pixels, un rectangle de 10x10 pixels est mis à l'échelle (soit 20x20 pixels). Si vous préférez utiliser une propriété `scaleMode` autre que `noScale` pour une scène de fenêtre, vous pouvez compenser le résultat en définissant le facteur d'échelle de tout objet d'affichage de la fenêtre sur le rapport 72 pixels/largeur et hauteur actuelles de la scène. Le code suivant calcule par exemple le facteur d'échelle requis de l'objet d'affichage `client` :

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

Remarque : *les fenêtres Flex et HTML définissent automatiquement la propriété `scaleMode` de la scène sur `noScale`. Modifier la propriété `scaleMode` entrave le fonctionnement des mécanismes de mise en forme automatique utilisés dans ces types de fenêtres.*

Utilisation du mode Plein écran

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le mode plein écran permet de définir la scène d'un film de sorte à remplir totalement le moniteur sans bordure ou menu. La propriété `displayState` de la classe Stage permet d'activer ou désactiver ce mode pour un fichier SWF. La propriété `displayState` peut être réglée sur l'une des valeurs définies par les constantes de la classe flash.display.StageDisplayState. Pour activer le mode plein écran, définissez la propriété `displayState` sur `StageDisplayState.FULL_SCREEN` :

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Pour activer le mode interactif plein écran (nouveauté dans Flash Player 11.3), définissez la propriété `displayState` sur `StageDisplayState.FULL_SCREEN_INTERACTIVE` :

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Dans Flash Player, le mode plein écran ne peut être activé que via ActionScript en réponse à un clic de souris (clic de bouton droit inclus) ou une frappe de touche. Le contenu AIR qui s'exécute dans le sandbox de sécurité de l'application ne nécessite pas que le mode plein écran soit activé en réponse à une action de l'utilisateur.

Pour quitter le mode plein écran, définissez la propriété `displayState` sur `StageDisplayState.NORMAL`.

```
stage.displayState = StageDisplayState.NORMAL;
```

Un utilisateur peut également désactiver le mode plein écran en plaçant le focus sur une autre fenêtre ou en utilisant l'une des combinaisons de touches suivantes : la touche Echap (toutes les plates-formes), Contrôle-W (Windows), Commande-W (Mac) ou Alt-F4 (Windows).

Activation du mode plein écran dans Flash Player

Pour activer le mode plein écran d'un fichier SWF intégré à une page HTML, le code HTML requis pour intégrer Flash Player doit comprendre une balise `param` et un attribut `embed`, associés au nom `allowFullScreen` et à la valeur `true`, comme suit :

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

Dans l'outil de programmation Flash, choisissez Fichier -> Paramètres de publication, puis dans la boîte de dialogue Paramètres de publication, cliquez sur l'onglet HTML et sélectionnez le modèle Flash seulement - Autorisation du Plein écran.

Dans Flex, assurez-vous que le modèle HTML inclut les balises `<object>` et `<embed>` qui prennent en charge le plein écran.

Si vous utilisez JavaScript dans une page Web pour générer les balises d'intégration de SWF, vous devez modifier le code JavaScript pour ajouter la balise et l'attribut `allowFullScreen param`. Par exemple, si votre page HTML fait appel à la fonction `AC_FL_RunContent()` (utilisée par les pages HTML générées par Flash Professional et Flash Builder), vous devez ajouter le paramètre `allowFullScreen` à cet appel de fonction, comme suit :

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

Ce cas de figure ne s'applique pas aux fichiers SWF qui s'exécutent dans la version autonome de Flash Player.

Remarque : *si vous définissez le Mode fenêtre (wmode dans le code HTML) sur Opaque sans fenêtre (opaque) ou Transparent sans fenêtre (transparent), la fenêtre plein écran est toujours opaque.*

Il existe également des restrictions liées à la sécurité lors de l'utilisation du mode plein écran avec Flash Player dans un navigateur. Ces restrictions sont décrites dans le chapitre « Sécurité » à la page 1085.

Activation du mode interactif plein écran dans Flash Player 11.3 et les versions ultérieures

Flash Player 11.3 et les versions ultérieures prennent en charge le mode interactif plein écran, qui permet une prise en charge totale de toutes les touches du clavier (à l'exception de la touche **Echap**, qui permet de quitter le mode interactif plein écran). Le mode interactif plein écran est utile pour les jeux (par exemple pour activer la fonction de dialogue en ligne dans un jeu multijoueurs ou les commandes du clavier WASD dans un jeu de tir subjectif.)

Pour activer le mode interactif plein écran d'un fichier SWF intégré à une page HTML, le code HTML requis pour intégrer Flash Player doit comprendre une balise `param` et un attribut `embed`, associés au nom `allowFullScreenInteractive` et à la valeur `true`, comme suit :

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

Dans l'outil de création de Flash, cliquez sur Fichier -> Paramètres de publication, puis, dans l'onglet HTML de la boîte de dialogue Paramètres de publication, sélectionnez le modèle Flash seulement - Autorisation du Plein écran interactif.

Dans Flash Builder et Flex, vérifiez que les modèles HTML incluent les balises `<object>` et `<embed>` qui prennent en charge le mode interactif plein écran.

Si vous utilisez JavaScript dans une page Web pour générer les balises d'intégration de SWF, vous devez modifier le code JavaScript pour ajouter la balise et l'attribut `allowFullScreenInteractive param`. Par exemple, si votre page HTML fait appel à la fonction `AC_FL_RunContent()` (utilisée par les pages HTML générées par Flash Professional et Flash Builder), vous devez ajouter le paramètre `allowFullScreenInteractive` à cet appel de fonction, comme suit :

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

Ce cas de figure ne s'applique pas aux fichiers SWF qui s'exécutent dans la version autonome de Flash Player.

Mise à l'échelle et taille de la scène en mode plein écran

Les propriétés `Stage.fullScreenHeight` et `Stage.fullScreenWidth` renvoient la hauteur et la largeur de l'écran utilisé lors de l'activation du mode plein écran, lorsque le passage à cet état est immédiat. Ces valeurs risquent d'être incorrectes si l'utilisateur déplace le navigateur d'un écran à un autre après avoir récupéré ces valeurs, mais avant de passer au mode plein écran. Si l'utilisateur récupère ces valeurs dans le gestionnaire d'événement dans lequel il a défini la propriété `Stage.displayState` sur `StageDisplayState.FULL_SCREEN`, celles-ci sont correctes. Pour les utilisateurs qui utilisent plusieurs écrans, le contenu du fichier SWF s'étend pour n'occuper qu'un seul écran. Flash Player et AIR utilisent une mesure pour identifier le moniteur contenant la portion la plus élevée du fichier SWF et activent le mode plein écran sur celui-ci. Les propriétés fullScreenHeight et fullScreenWidth ne reflètent que la taille du moniteur utilisé en mode plein écran. Pour plus d'informations, voir `Stage.fullScreenHeight` et `Stage.fullScreenWidth` dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

En mode plein écran, le comportement de mise à l'échelle de la scène est identique à celui du mode normal. Cette mise à l'échelle est contrôlée par la propriété `scaleMode` de la classe Stage. Si la propriété `scaleMode` est définie sur `StageScaleMode.NO_SCALE`, les propriétés `stageWidth` et `stageHeight` de la classe Stage sont modifiées pour répercuter la taille de la zone de l'écran occupée par le fichier SWF (soit, dans ce cas, l'écran entier). Dans un navigateur, le paramètre HTML correspondant contrôle le réglage.

L'événement `fullScreen` de la classe Stage permet de détecter et répondre à l'activation ou à la désactivation du mode plein écran. Par exemple, il peut être nécessaire de repositionner, d'ajouter ou de supprimer des éléments lors d'un changement d'état du mode plein écran, comme illustré par cet exemple :

```
import flash.events.FullScreenEvent;

function fullScreenRedraw(event:FullScreenEvent):void
{
    if (event.fullScreen)
    {
        // Remove input text fields.
        // Add a button that closes full-screen mode.
    }
    else
    {
        // Re-add input text fields.
        // Remove the button that closes full-screen mode.
    }
}

mySprite.stage.addEventListener(FullScreenEvent.FULL_SCREEN, fullScreenRedraw);
```

Comme l'indique ce code, l'objet associé à l'événement `fullScreen` est une occurrence de la classe flash.events.FullScreenEvent, dont la propriété `fullScreen` indique si le mode plein écran est activé (`true`) ou non (`false`).

Prise en charge du clavier en mode plein écran

Lorsque Flash Player est exécuté dans un navigateur, l'ensemble du code ActionScript lié au clavier (événements de clavier et saisie de texte dans des occurrences de TextField, entre autres), est désactivé en mode plein écran. Les exceptions (c'est-à-dire les touches qui restent actives) sont les suivantes :

- Les touches hors impression sélectionnées, notamment les touches fléchées, la barre d'espacement et la touche de tabulation
- Les raccourcis clavier qui désactivent le mode plein écran : touche Echap (Windows et Mac), Contrôle-W (Windows), Commande-W (Mac) et Alt-F4

Ces restrictions ne sont pas présentes pour le contenu SWF s'exécutant dans l'application Flash Player autonome ou dans AIR. AIR prend en charge un mode plein écran interactif qui permet la saisie clavier.

Prise en charge de la souris en mode plein écran

Par défaut, les événements de souris fonctionnent de la même façon en mode plein écran que dans un autre mode d'affichage. Néanmoins, le mode plein écran permet également de définir la propriété `Stage.mouseLock` en vue de verrouiller la souris. Le verrouillage de la souris désactive le curseur et permet de déplacer la souris sans aucune limite.

Remarque : *vous pouvez verrouiller la souris uniquement sur des applications de bureau en mode plein écran. Le verrouillage de la souris sur des applications dans un autre mode d'affichage ou sur des applications mobiles renvoie une exception.*

Le verrouillage de la souris est automatiquement désactivé et le curseur de la souris est à nouveau visible lorsque :

- l'utilisateur quitte le mode plein écran en appuyant sur la touche Echap (toutes les plates-formes), ou sur les touches Ctrl-W (Windows), Cmd-W (Mac) ou Alt-F4 (Windows) ;
- la fenêtre de l'application perd le focus ;
- une interface de paramétrage est visible, notamment une boîte de dialogue de confidentialité ;
- une boîte de dialogue native s'affiche, notamment une boîte de dialogue de téléchargement d'un fichier.

Les événements associés au mouvement de la souris, tels que l'événement `mouseMove`, font appel à la classe MouseEvent pour représenter l'objet de l'événement. Lorsque le verrouillage de la souris est désactivé, utilisez les propriétés `MouseEvent.localX` et `MouseEvent.localY` pour déterminer l'emplacement de la souris. Lorsque le verrouillage de la souris est activé, utilisez les propriétés `MouseEvent.movementX` et `MouseEvent.movementY` pour déterminer l'emplacement de la souris. Les propriétés `movementX` et `movementY` contiennent les changements de position de la souris depuis le dernier événement plutôt que les coordonnées absolues de l'emplacement de la souris.

Mise à l'échelle matérielle en mode plein écran

La propriété `fullScreenSourceRect` de la classe Stage permet d'utiliser Flash Player ou AIR pour mettre à l'échelle une zone déterminée de la scène en mode plein écran. Dans Flash Player et AIR, la mise à l'échelle matérielle, si elle est disponible, s'effectue à l'aide de la carte graphique et de la carte vidéo de l'ordinateur, et permet généralement d'afficher le contenu plus rapidement que la mise à l'échelle logicielle.

Pour tirer parti de la mise à l'échelle matérielle, définissez l'ensemble ou une partie de la scène sur le mode plein écran. Le code suivant ActionScript 3.0 définit l'ensemble de la scène en mode plein écran :

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Lorsque cette propriété est définie sur un rectangle valide et la propriété `displayState` sur le mode plein écran, Flash Player et AIR redimensionnent la zone spécifiée. La taille réelle de la scène en pixels dans ActionScript ne change pas. Flash Player et AIR imposent une taille limite au rectangle en fonction de la taille du message standard « Appuyez sur la touche Echap pour quitter le mode plein écran ». Cette limite est généralement d'environ 260 sur 30 pixels, mais peut varier en fonction de la plate-forme et de la version de Flash Player.

La propriété `fullScreenSourceRect` ne peut être définie que lorsque Flash Player ou AIR ne sont pas en mode plein écran. Pour utiliser correctement cette propriété, vous devez tout d'abord la définir, puis définir la propriété `displayState` sur le mode plein écran.

Pour activer la mise à l'échelle, définissez la propriété `fullScreenSourceRect` sur un objet rectangle.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

Pour désactiver la mise à l'échelle, définissez la propriété `fullScreenSourceRect` sur `null`.

```
stage.fullScreenSourceRect = null;
```

Pour tirer parti de toutes les fonctions d'accélération matérielle avec Flash Player, activez cette fonction dans la boîte de dialogue des paramètres de Flash Player. Pour afficher cette boîte de dialogue, cliquez avec le bouton droit de la souris (Windows) ou cliquez tout en appuyant sur la touche Ctrl (Mac) dans le contenu Flash Player dans votre navigateur. Cliquez sur le premier onglet, Affichage, puis cochez la case Activer l'accélération matérielle.

Modes Fenêtre direct et composition GPU

Flash Player 10 propose deux modes de fenêtre, direct et composition GPU, que vous pouvez activer dans les paramètres de publication de l'outil de programmation Flash. Ces modes ne sont pas pris en charge par AIR. Pour tirer parti de ces modes, vous devez activer l'accélération matérielle pour Flash Player.

Le mode direct utilise le chemin le plus rapide et le plus direct pour placer des graphismes sur l'écran, ce qui est pratique pour la lecture vidéo.

La composition GPU utilise l'unité de traitement graphique sur la carte vidéo pour accélérer la composition. La composition vidéo consiste à créer des images multi-couches pour aboutir à une seule image vidéo. Lorsque la composition est accélérée par la GPU, les performances de la conversion YUV, la correction des couleurs, la rotation, le redimensionnement et le fondu sont nettement améliorés. La conversion YUV se réfère à la conversion des couleurs de signaux analogiques composites, qui sont utilisées pour la transmission, au modèle de couleurs RVB (rouge-vert-bleu) que les caméras vidéo et les écrans utilisent. Le recours à la GPU pour accélérer la composition réduit la demande en mémoire et en puissance de calcul qui affectera les performances de l'unité centrale. Il en résulte une lecture vidéo plus régulière en définition standard.

Soyez vigilant lorsque vous implémentez ces modes Fenêtre. Fenêtre L'utilisation de la composition GPU peut s'avérer coûteuse en mémoire et en ressources de l'unité centrale. Si certaines opérations (comme les modes fondu, le filtrage, l'écrêtage ou le masquage) ne peuvent pas être exécutées dans la GPU, c'est le logiciel qui s'en charge. Adobe vous recommande de vous limiter à un fichier SWF par page HTML lorsque vous utilisez ces modes. Il ne faudrait pas les utiliser pour des bandeaux. La Flash Test Movie facility ne fait pas appel à l'accélération matérielle mais vous pouvez l'utiliser par le biais de l'option Aperçu avant publication.

Il est inutile de fixer une cadence supérieure à 60 (c'est la fréquence de régénération d'écran maximale) dans votre fichier SWF. Une cadence entre 50 et 55 débouche sur des images perdues, ce qui peut se produire de temps à autre pour des raisons diverses.

Le mode direct requiert Microsoft DirectX 9 avec une mémoire virtuelle RAM de 128 Mo sous Windows et OpenGL pour Apple Macintosh, Mac OS X version 10.2 ou ultérieure. La composition GPU requiert Microsoft DirectX 9 et la prise en charge de Pixel Shader 2.0 sous Windows avec une mémoire virtuelle RAM de 128 Mo. Sous Mac OS X et Linux, la composition GPU requiert OpenGL 1.5 et plusieurs extensions d'OpenGL (objet framebuffer, objets multitexture et shader, langage d'ombrage, shader de fragments).

Vous pouvez activer les modes d'accélération `direct` et `gpu` pour chaque SWF par le biais de la boîte de dialogue Paramètres de publication de Flash, à l'aide du menu d'accélération matérielle sur l'onglet Flash. Si vous choisissez Aucun, le mode Fenêtre revient à `défaut`, `transparent` ou `opaque`, selon le paramètre du Mode Fenêtre de l'onglet HTML.

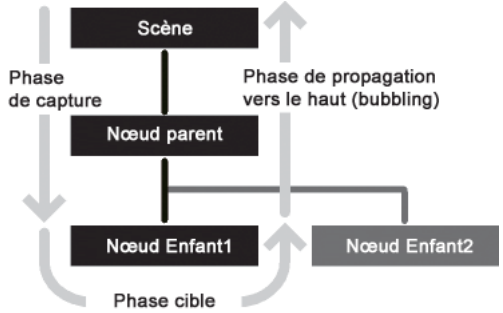
Manipulation des événements associés aux objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe DisplayObject hérite de la classe EventDispatcher. Cela signifie que chaque objet d'affichage peut être pleinement impliqué dans le modèle d'événement (décrit dans le chapitre « Gestion des événements » à la page 129). Chaque objet d'affichage peut utiliser sa méthode `addEventListener()` (héritée de la classe EventDispatcher) pour attendre un événement particulier, mais ceci uniquement si l'objet écouteur fait partie du flux d'événement de l'événement considéré.

Lorsque Flash Player ou AIR distribue un objet événement, celui-ci effectue un aller-retour à partir de la scène via l'objet d'affichage où s'est produit l'événement. Par exemple, si un utilisateur clique sur un objet d'affichage appelé `child1`, Flash Player distribue un objet événement à partir de la scène via la hiérarchie de la liste d'affichage jusqu'à l'objet d'affichage `child1`.

D'un point de vue conceptuel, le flux d'événement est divisé en trois phases, comme illustré par le diagramme suivant :



Pour plus d'informations, voir « Gestion des événements » à la page 129.

Lors de la gestion des événements liés aux objets d'affichage, il est important de ne pas oublier l'effet potentiel des objets écouteurs d'événement sur l'éventuelle suppression automatique des objets d'affichage de la mémoire (garbage collection) lorsqu'ils sont supprimés de la liste d'affichage. Si des objets d'un objet d'affichage sont enregistrés en tant qu'écouteurs d'événement, ce dernier n'est pas supprimé de la mémoire même s'il est supprimé de la liste d'affichage, car il continue à posséder des références à ces objets écouteur. Pour plus d'informations, voir « Gestion des écouteurs d'événement » à la page 143.

Sélection d'une sous-classe de DisplayObject

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Plusieurs options étant disponibles, l'une des décisions importantes à prendre lors de l'utilisation d'objets d'affichage est le choix de l'objet à utiliser dans un but précis. Les directives suivantes vous aideront à choisir l'option appropriée. Ces suggestions sont valides que vous créiez une occurrence de classe ou que vous choisissiez une classe de base pour une classe en cours de création :

- Si vous n'avez pas besoin d'un objet pouvant contenir d'autres objets d'affichage (autrement dit, si vous avez simplement besoin d'un objet à utiliser comme élément isolé), choisissez l'une des sous-classes suivantes de DisplayObject ou d'InteractiveObject, selon l'utilisation prévue :
  - Bitmap, pour afficher une image bitmap.
  - TextField, pour ajouter du texte.
  - Video, pour afficher une vidéo.
  - Shape, pour disposer d'une « toile » destinée au tracé d'un contenu à l'écran. En particulier si vous souhaitez créer une occurrence pour dessiner des formes à l'écran et qu'elle ne contient pas d'autres objets d'affichage, vous obtiendrez des performances nettement supérieures en utilisant Shape plutôt que Sprite ou MovieClip.
  - MorphShape, StaticText ou SimpleButton pour des éléments créés par l'outil de programmation Flash. Il est impossible de créer par programmation des occurrences de ces classes, mais vous pouvez créer des variables avec ces types de données pour pointer sur des éléments créés dans l'outil de programmation Flash.
- Si vous devez disposer d'une variable qui se réfère à la scène principale, utilisez la classe Stage en tant que type de données.
- Si vous devez disposer d'un conteneur pour charger un fichier SWF ou fichier image externe, utilisez une occurrence de Loader. Le contenu chargé sera ajouté à la liste d'affichage en tant qu'enfant de l'occurrence de Loader. Son type de données variera selon la nature du contenu chargé, comme suit :
  - Une image chargée est une occurrence de Bitmap.
  - Un fichier SWF chargé écrit dans ActionScript 3.0 est une occurrence de Sprite ou MovieClip (ou une occurrence d'une sous-classe de ces classes, comme indiqué par le créateur du contenu).
  - Un fichier SWF chargé écrit dans ActionScript 1.0 ou ActionScript 2.0 est une occurrence d'AVM1Movie.
- Si vous avez besoin d'un objet qui sera le conteneur d'autres objets d'affichage (que vous ayez ou non l'intention de dessiner en ActionScript dans cet objet), choisissez l'une des sous-classes de DisplayObjectContainer :
  - Sprite, si l'objet est créé en ActionScript uniquement ou en tant que classe de base d'un objet d'affichage personnalisé créé et manipulé en ActionScript uniquement.
  - MovieClip, si vous créez une variable pointant vers un symbole de clip créé dans l'outil de programmation Flash.
- Si vous créez une classe qui sera associée à un symbole de clip de la bibliothèque de Flash, choisissez l'une des sous-classes de DisplayObjectContainer comme classe de base de votre future classe :
  - MovieClip, si le symbole de clip associé possède un contenu sur plusieurs images.
  - Sprite, si le symbole de clip associé ne possède un contenu que sur la première image.

Manipulation des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Quel que soit l'objet d'affichage utilisé, tous les éléments affichés à l'écran partagent diverses techniques de manipulations. Par exemple, ils peuvent tous être positionnés à l'écran, avancer ou reculer dans l'ordre d'empilement des objets d'affichage, mis à l'échelle, pivotés, et ainsi de suite. Dans la mesure où tous les objets d'affichage héritent ces fonctionnalités de leur classe de base commune (DisplayObject), lesdites fonctionnalités ont le même comportement pour une occurrence d'un objet TextField, Video, Shape ou tout autre objet d'affichage. Les sections suivantes passent en revue plusieurs de ces manipulations appliquées à tous les objets d'affichage.

Modification de la position

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La manipulation la plus fondamentale de tout objet d'affichage consiste à le positionner à l'écran. Pour définir la position d'un objet d'affichage, changez ses propriétés `x` et `y`.

```
myShape.x = 17;
myShape.y = 212;
```

Le système de positionnement d'un objet d'affichage assimile la scène à un système de coordonnées cartésiennes (système de grille standard doté d'un axe horizontal x et d'un axe vertical y). L'origine du système de coordonnées (coordonnée 0,0 correspondant à l'intersection des axes x et y) figure dans le coin supérieur gauche de la scène. A partir de l'origine, les valeurs x sont positives vers la droite et négatives vers la gauche, tandis que, contrairement aux systèmes de graphes standard, les valeurs y sont positives vers le bas et négatives vers le haut. Par exemple, les lignes précédentes de code déplacent l'objet `myShape` vers la coordonnée x 17 (17 pixels sur la droite de l'origine) et la coordonnée y 212 (212 pixels sous l'origine).

Par défaut, lorsqu'un objet d'affichage est créé dans ActionScript, les propriétés `x` et `y` sont toutes deux définies sur 0, plaçant ainsi l'objet dans le coin supérieur gauche de son contenu parent.

Modification de la position relativement à la scène

Il est important de se rappeler que les propriétés `x` et `y` se réfèrent toujours à la position de l'objet d'affichage par rapport aux coordonnées 0,0 des axes de son objet d'affichage parent. Ainsi, pour une occurrence de Shape (par exemple un cercle) contenue dans une occurrence de Sprite, mettre à zéro les propriétés `x` et `y` de l'objet Shape revient à placer le cercle dans le coin supérieur gauche de l'objet Sprite, qui n'est pas forcément le coin supérieur gauche de la scène. Pour positionner un objet par rapport aux coordonnées globales de la scène, utilisez la méthode `globalToLocal()` de tout objet d'affichage afin de convertir les coordonnées globales (scène) en coordonnées locales (conteneur de l'objet d'affichage), comme suit :

```
// Position the shape at the top-left corner of the Stage,
// regardless of where its parent is located.

// Create a Sprite, positioned at x:200 and y:200.
var mySprite:Sprite = new Sprite();
mySprite.x = 200;
mySprite.y = 200;
this.addChild(mySprite);

// Draw a dot at the Sprite's 0,0 coordinate, for reference.
mySprite.graphics.lineStyle(1, 0x000000);
mySprite.graphics.beginFill(0x000000);
mySprite.graphics.moveTo(0, 0);
mySprite.graphics.lineTo(1, 0);
mySprite.graphics.lineTo(1, 1);
mySprite.graphics.lineTo(0, 1);
mySprite.graphics.endFill();

// Create the circle Shape instance.
var circle:Shape = new Shape();
mySprite.addChild(circle);

// Draw a circle with radius 50 and center point at x:50, y:50 in the Shape.
circle.graphics.lineStyle(1, 0x000000);
circle.graphics.beginFill(0xff0000);
circle.graphics.drawCircle(50, 50, 50);
circle.graphics.endFill();

// Move the Shape so its top-left corner is at the Stage's 0, 0 coordinate.
var stagePoint:Point = new Point(0, 0);
var targetPoint:Point = mySprite.globalToLocal(stagePoint);
circle.x = targetPoint.x;
circle.y = targetPoint.y;
```

Vous pouvez également utiliser la méthode `localToGlobal()` de la classe DisplayObject pour convertir les coordonnées locales en coordonnées de la scène.

Déplacement des objets d'affichage à l'aide de la souris

Vous pouvez autoriser un utilisateur à déplacer les objets d'affichage à l'aide de la souris par le biais de deux techniques distinctes dans ActionScript. Dans les deux cas, deux événements de souris sont utilisés : lorsque l'utilisateur appuie sur le bouton de gauche (l'objet reçoit alors l'instruction de suivre le curseur de la souris) et lorsqu'il le relâche (l'objet reçoit alors l'instruction de cesser de suivre le curseur de la souris).

Remarque : *Flash Player 11.3 et les versions ultérieures et AIR 3.3 et les versions ultérieures : vous pouvez également utiliser l'événement MouseEvent.RELEASE_OUTSIDE dans le cas d'un utilisateur qui relâche le bouton de la souris en dehors des limites du Sprite conteneur.*

La première technique, basée sur la méthode `startDrag()`, est plus simple, mais plus limitée. Lorsque l'utilisateur appuie sur le bouton de la souris, la méthode `startDrag()` de l'objet d'affichage à faire glisser est appelée. Lorsque l'utilisateur relâche le bouton de la souris, la méthode `stopDrag()` est appelée. Puisque la classe Sprite définit ces deux fonctions, l'objet déplacé doit être un Sprite ou l'une de ses sous-classes.

```
// This code creates a mouse drag interaction using the startDrag()
// technique.
// square is a MovieClip or Sprite instance).

import flash.events.MouseEvent;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    square.startDrag();
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    square.stopDrag();
}

square.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
square.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Cette technique présente un désavantage relativement important : l'utilisation de `startDrag()` ne permet de faire glisser qu'un seul élément à la fois. Si un objet d'affichage est en cours de glissement et que la méthode `startDrag()` est appelée sur un autre objet d'affichage, le premier objet cesse immédiatement de suivre la souris. Par exemple, si la fonction `startDragging()` est modifiée comme indiqué, seul l'objet `circle` est déplacé par glissement, bien que la méthode `square.startDrag()` ait été appelée :

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

Puisqu'un seul objet à la fois peut être déplacé par glissement par le biais de `startDrag()`, la méthode `stopDrag()` peut être appelée sur n'importe quel objet d'affichage et arrête tout objet en cours de glissement.

Pour déplacer plusieurs objets d'affichage, ou pour éviter tout risque de conflit au cas où plusieurs objets seraient susceptibles d'utiliser `startDrag()`, il est préférable d'utiliser la technique de suivi de la souris pour créer l'effet de glisser-déposer. Avec cette technique, lorsque l'utilisateur appuie sur le bouton de la souris, une fonction est enregistrée en tant qu'écouteur de l'événement `mouseMove` de la scène. Cette fonction, qui est alors appelée à chaque déplacement de la souris, entraîne le saut de l'objet déplacé vers la coordonnée x, y de la souris. Une fois le bouton de la souris relâché, la fonction n'est plus enregistrée en tant qu'écouteur. Elle n'est donc plus appelée lorsque la souris est déplacée et l'objet cesse de suivre le curseur. Le code suivant illustre cette technique :

```
// This code moves display objects using the mouse-following
// technique.
// circle is a DisplayObject (e.g. a MovieClip or Sprite instance).

import flash.events.MouseEvent;

var offsetX:Number;
var offsetY:Number;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    // Record the difference (offset) between where
    // the cursor was when the mouse button was pressed and the x, y
    // coordinate of the circle when the mouse button was pressed.
    offsetX = event.stageX - circle.x;
    offsetY = event.stageY - circle.y;

    // tell Flash Player to start listening for the mouseMove event
    stage.addEventListener(MouseEvent.MOUSE_MOVE, dragCircle);
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    // Tell Flash Player to stop listening for the mouseMove event.
    stage.removeEventListener(MouseEvent.MOUSE_MOVE, dragCircle);
}

// This function is called every time the mouse moves,
// as long as the mouse button is pressed down.
function dragCircle(event:MouseEvent):void
{
    // Move the circle to the location of the cursor, maintaining
    // the offset between the cursor's location and the
    // location of the dragged object.
    circle.x = event.stageX - offsetX;
    circle.y = event.stageY - offsetY;

    // Instruct Flash Player to refresh the screen after this event.
    event.updateAfterEvent();
}

circle.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
circle.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Outre le suivi du curseur par un objet d'affichage, il est souvent préférable de positionner l'objet déplacé à l'avant de l'affichage, créant ainsi une impression de flottement au-dessus de tous les autres objets. Supposons, par exemple, que deux objets, un cercle et un carré, puissent tous deux être déplacés à l'aide de la souris. Si le cercle figure sous le carré sur la liste d'affichage et que vous cliquez et faites glisser le cercle pour placer le curseur au-dessus du carré, il semble glisser derrière le carré, brisant ainsi l'illusion du glisser-déposer. Vous pouvez procéder de sorte que lorsque vous cliquez sur le cercle, il passe en première position dans la liste d'affichage et s'affiche ainsi par-dessus tout autre contenu.

Le code suivant (adapté de l'exemple précédent) permet de déplacer deux objets d'affichage, un cercle et un carré, à l'aide de la souris. Lorsque le bouton de la souris est pressé au-dessus de l'un d'eux, cet élément est amené en haut de la liste d'affichage de la scène, afin que l'élément déplacé passe toujours devant les autres. (Tout nouveau code ou code modifié extrait du code précédent est imprimé en gras.)

```
// This code creates a drag-and-drop interaction using the mouse-following
// technique.
// circle and square are DisplayObjects (e.g. MovieClip or Sprite
// instances).

import flash.display.DisplayObject;
import flash.events.MouseEvent;

var offsetX:Number;
var offsetY:Number;
var draggedObject:DisplayObject;

// This function is called when the mouse button is pressed.
function startDragging(event:MouseEvent):void
{
    // remember which object is being dragged
    draggedObject = DisplayObject(event.target);

    // Record the difference (offset) between where the cursor was when
    // the mouse button was pressed and the x, y coordinate of the
    // dragged object when the mouse button was pressed.
    offsetX = event.stageX - draggedObject.x;
    offsetY = event.stageY - draggedObject.y;

    // move the selected object to the top of the display list
    stage.addChild(draggedObject);

    // Tell Flash Player to start listening for the mouseMove event.
    stage.addEventListener(MouseEvent.MOUSE_MOVE, dragObject);
}

// This function is called when the mouse button is released.
function stopDragging(event:MouseEvent):void
{
    // Tell Flash Player to stop listening for the mouseMove event.
    stage.removeEventListener(MouseEvent.MOUSE_MOVE, dragObject);
}
```

```
// This function is called every time the mouse moves,
// as long as the mouse button is pressed down.
function dragObject(event:MouseEvent):void
{
    // Move the dragged object to the location of the cursor, maintaining
    // the offset between the cursor's location and the location
    // of the dragged object.
    draggedObject.x = event.stageX - offsetX;
    draggedObject.y = event.stageY - offsetY;

    // Instruct Flash Player to refresh the screen after this event.
    event.updateAfterEvent();
}

circle.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
circle.addEventListener(MouseEvent.MOUSE_UP, stopDragging);

square.addEventListener(MouseEvent.MOUSE_DOWN, startDragging);
square.addEventListener(MouseEvent.MOUSE_UP, stopDragging);
```

Pour créer un effet plus sophistiqué, par exemple pour un jeu où des jetons ou des cartes passent d'une pile à l'autre, il est possible d'ajouter l'objet déplacé à la liste d'affichage de la scène lorsqu'il est « pris », puis de l'ajouter à une autre liste d'affichage (la « pile » sur laquelle il est déposé) lors du relâchement du bouton de souris.

Enfin, pour optimiser l'effet, vous pourriez appliquer un filtre Ombre portée à l'objet d'affichage lorsque vous cliquez dessus (en début de glissement) et supprimer l'ombre portée lorsque vous relâchez l'objet. Pour plus d'informations sur le filtre Ombre portée et tout autre filtre appliqué aux objets d'affichage en ActionScript, voir « Filtrage des objets d'affichage » à la page 276.

Défilement horizontal ou vertical des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si la taille d'un objet d'affichage est trop élevée pour la zone où vous souhaitez l'afficher, vous disposez de la propriété `scrollRect` pour définir la zone visible de l'objet d'affichage. Par ailleurs, en modifiant la propriété `scrollRect` en réponse à une action de l'utilisateur, vous pouvez entraîner un défilement vers la gauche ou la droite ou vers le haut ou le bas.

La propriété `scrollRect` est une occurrence de la classe Rectangle, qui combine les valeurs requises pour définir une zone rectangulaire en tant qu'objet unique. Pour définir initialement la zone visible de l'objet d'affichage, créez une occurrence de Rectangle et affectez-la à la propriété `scrollRect` de l'objet. Par la suite, pour obtenir un défilement horizontal ou vertical, il suffit de lire la propriété `scrollRect` dans une variable Rectangle séparée et de changer la propriété voulue (par exemple, modifier la propriété `x` de l'occurrence de Rectangle pour un défilement horizontal, ou sa propriété `y` pour un défilement vertical). Vous réaffectez ensuite cette occurrence de Rectangle à la propriété `scrollRect` pour avertir l'objet d'affichage du changement de valeur.

Par exemple, le code suivant définit la zone visible d'un objet TextField nommé `bigText` dont la hauteur est trop importante pour les dimensions du fichier SWF. Lorsque l'utilisateur clique sur les deux boutons nommés `up` et `down`, les fonctions appelées entraînent le défilement vertical du contenu de l'objet TextField en modifiant la propriété `y` de l'occurrence de Rectangle `scrollRect`.

```
import flash.events.MouseEvent;
import flash.geom.Rectangle;

// Define the initial viewable area of the TextField instance:
// left: 0, top: 0, width: TextField's width, height: 350 pixels.
bigText.scrollRect = new Rectangle(0, 0, bigText.width, 350);

// Cache the TextField as a bitmap to improve performance.
bigText.cacheAsBitmap = true;

// called when the "up" button is clicked
function scrollUp(event:MouseEvent):void
{
    // Get access to the current scroll rectangle.
    var rect:Rectangle = bigText.scrollRect;
    // Decrease the y value of the rectangle by 20, effectively
    // shifting the rectangle down by 20 pixels.
    rect.y -= 20;
    // Reassign the rectangle to the TextField to "apply" the change.
    bigText.scrollRect = rect;
}

// called when the "down" button is clicked
function scrollDown(event:MouseEvent):void
{
    // Get access to the current scroll rectangle.
    var rect:Rectangle = bigText.scrollRect;
    // Increase the y value of the rectangle by 20, effectively
    // shifting the rectangle up by 20 pixels.
    rect.y += 20;
    // Reassign the rectangle to the TextField to "apply" the change.
    bigText.scrollRect = rect;
}

up.addEventListener(MouseEvent.CLICK, scrollUp);
down.addEventListener(MouseEvent.CLICK, scrollDown);
```

Comme le montre cet exemple, lorsque vous utilisez la propriété `scrollRect` d'un objet d'affichage, il est préférable de spécifier que Flash ou AIR doit mettre en cache le contenu de l'objet sous forme de bitmap, à l'aide de la propriété `cacheAsBitmap`. Ainsi, Flash Player et AIR n'ont pas à redessiner le contenu complet de l'objet d'affichage à chaque défilement de celui-ci, et peuvent utiliser le bitmap mis en cache pour afficher la portion concernée directement à l'écran. Pour plus d'informations, voir « Mise en cache des objets d'affichage » à la page 189.

Manipulation de la taille et de l'échelle des objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La taille d'un objet d'affichage peut être obtenue et modifiée de deux façons, en utilisant soit les propriétés de dimensions (`width` et `height`), soit les propriétés d'échelle (`scaleX` et `scaleY`).

Chaque objet d'affichage possède une propriété `width` et une propriété `height`, qui sont initialement définies sur la taille de l'objet en pixels. Vous pouvez lire les valeurs de ces propriétés pour mesurer la taille de l'objet d'affichage. Vous pouvez également stipuler de nouvelles valeurs pour modifier la taille de l'objet, comme suit :

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

Modifier les propriétés `height` ou `width` d'un objet d'affichage modifie l'échelle de ce dernier. En d'autres termes, son contenu est étiré ou comprimé en fonction de la taille de la nouvelle zone. Si l'objet d'affichage ne contient que des formes vectorielles, elles sont redessinées à la nouvelle échelle, sans perte de qualité. Tout élément graphique bitmap de l'objet d'affichage est mis à l'échelle au lieu d'être redessiné. Ainsi, une photo numérique dont la largeur et la hauteur augmentent de sorte à dépasser les dimensions réelles des informations relatives aux pixels de l'image est pixellisée, ce qui lui donne un aspect irrégulier.

Lorsque vous modifiez les propriétés `width` ou `height` d'un objet d'affichage, Flash Player et AIR mettent également à jour les propriétés `scaleX` ou `scaleY` de l'objet.

Remarque : *les objets TextField ne respectent pas ce comportement de mise à l'échelle. Les champs de texte doivent se redimensionner automatiquement pour gérer le retour à la ligne automatique et les tailles de police. Leurs valeurs scaleX et scaleY sont donc réinitialisées à 1 au terme du redimensionnement. Toutefois, si vous ajustez la valeur scaleX ou scaleY d'un objet TextField, les valeurs de largeur et de hauteur sont modifiées en fonction des valeurs de mise à l'échelle que vous indiquez.*

Ces propriétés représentent la taille relative de l'objet d'affichage par rapport à sa taille d'origine. Les propriétés `scaleX` et `scaleY` utilisent des valeurs exprimées sous forme de fractions (décimales) pour représenter le pourcentage. Par exemple, si la propriété `width` d'un objet d'affichage a été réduite à la moitié de sa largeur originale, la propriété `scaleX` de cet objet prendra la valeur `0,5`, soit 50 %. Si sa hauteur a doublé, sa propriété `scaleY` prendra la valeur `2`, soit 200 %.

```
// circle is a display object whose width and height are 150 pixels.
// At original size, scaleX and scaleY are 1 (100%).
trace(circle.scaleX); // output: 1
trace(circle.scaleY); // output: 1

// When you change the width and height properties,
// Flash Player changes the scaleX and scaleY properties accordingly.
circle.width = 100;
circle.height = 75;
trace(circle.scaleX); // output: 0.6622516556291391
trace(circle.scaleY); // output: 0.4966887417218543
```

Les changements de taille ne sont pas proportionnels. En d'autres termes, si vous modifiez la `hauteur` d'un carré, mais non sa `largeur`, ses proportions ne sont plus identiques et il devient un rectangle au lieu d'un carré. Si vous souhaitez modifier relativement la taille d'un objet d'affichage, vous pouvez définir les valeurs des propriétés `scaleX` et `scaleY` pour redimensionner l'objet, plutôt que définir les propriétés `width` ou `height`. Par exemple, ce code modifie la propriété `width` de l'objet d'affichage appelé `square`, puis modifie l'échelle verticale (`scaleY`) en fonction de l'échelle horizontale, afin que la taille du carré demeure proportionnelle.

```
// Change the width directly.
square.width = 150;

// Change the vertical scale to match the horizontal scale,
// to keep the size proportional.
square.scaleY = square.scaleX;
```

Contrôle de la distorsion lors de la mise à l'échelle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En règle générale, lorsqu'un objet d'affichage est mis à l'échelle (s'il est étiré horizontalement, par exemple), la distorsion résultante est répartie équitablement dans l'objet, de sorte que chaque partie soit soumise à un étirement identique. Pour les graphiques et éléments de conception, cette technique correspond probablement au résultat escompté. Toutefois, il est parfois préférable de garder le contrôle sur les parties de l'objet qui seront étirées et celles qui ne le seront pas. Un exemple courant est celui d'un bouton représenté par un rectangle aux angles arrondis. Lors d'une mise à l'échelle normale, les angles du bouton sont étirés, entraînant ainsi la modification de leur rayon lorsque le bouton est redimensionné.

Mais dans ce cas précis, il serait préférable de contrôler la mise à l'échelle, c'est-à-dire de pouvoir désigner certaines zones qui seront mises à l'échelle (les côtés) et celles qui ne le seront pas (les angles), afin que le changement d'échelle ne provoque pas de distorsion visible.

Vous disposez de la mise à l'échelle à 9 découpes (Echelle-9) pour créer des objets d'affichage dont vous contrôlez le mode de mise à l'échelle. La mise à l'échelle à 9 découpes permet de diviser l'objet d'affichage en neuf rectangles distincts (grille de 3 sur 3). Ces rectangles ne sont pas obligatoirement de la même taille, car l'utilisateur choisit l'emplacement des lignes de séparation. Tout contenu placé dans les quatre rectangles de coin (tels que les angles arrondis d'un bouton) n'est ni étiré, ni comprimé lors de la mise à l'échelle de l'objet d'affichage. Les rectangles placés en haut au centre et en bas au centre sont mis à l'échelle horizontalement, mais non verticalement, tandis que les rectangles centraux de gauche et de droite sont mis à l'échelle verticalement, mais non horizontalement. Le rectangle central est mis à l'échelle horizontalement et verticalement.

Ainsi, si vous créez un objet d'affichage et souhaitez qu'une partie du contenu ne subisse jamais de mise à l'échelle, il vous suffit de placer les lignes de séparation de la grille de mise à l'échelle à 9 découpes de telle sorte que ce contenu se trouve dans l'un des rectangles des angles.

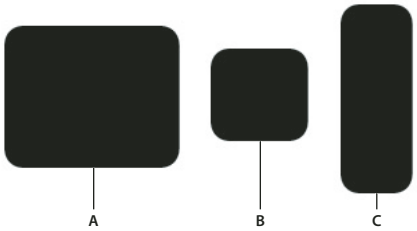
En ActionScript, il suffit de définir une valeur pour la propriété `scale9Grid` d'un objet d'affichage pour activer la mise à l'échelle à 9 découpes pour cet objet et définir la taille des rectangles de la grille. Vous utilisez une occurrence de la classe Rectangle en tant que valeur de la propriété `scale9Grid`, comme suit :

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

Les quatre paramètres du constructeur Rectangle sont la coordonnée x, la coordonnée y, la largeur et la hauteur. Dans cet exemple, l'angle supérieur gauche du rectangle est placé au point x : 32, y : 27 sur l'objet d'affichage appelé `myButton`. Le rectangle mesure 71 pixels de large et 64 pixels de haut (son bord droit correspond à la coordonnée x 103 sur l'objet d'affichage et son bord inférieur à la coordonnée y 92 sur l'objet d'affichage).

La zone figurant dans la région définie par l'occurrence de Rectangle représente le rectangle central de la grille Echelle-9. Les autres rectangles sont calculés par Flash Player et AIR en prolongeant les côtés de l'occurrence de Rectangle, comme indiqué :

Dans ce cas de figure, lorsque la mise à l'échelle du bouton augmente ou diminue, les angles arrondis ne sont ni étirés, ni comprimés, mais les autres zones sont ajustées en conséquence.



A. *myButton.width = 131;myButton.height = 106;* ***B.*** *myButton.width = 73;myButton.height = 69;* ***C.*** *myButton.width = 54;myButton.height = 141;*

Mise en cache des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Que vous créiez une application ou des animations scriptées complexes, vous devez considérer la performance et l'optimisation, à mesure que la taille de vos conceptions Flash augmente. Lorsque votre contenu demeure statique (par exemple une occurrence de Shape rectangulaire), Flash et AIR ne l'optimisent pas. Par conséquent, lorsque vous modifiez la position du rectangle, Flash Player ou AIR redessine la totalité de l'occurrence de Shape.

Vous pouvez mettre en cache des objets d'affichage spécifiques pour améliorer les performances de votre fichier SWF. L'objet d'affichage est essentiellement une *surface*, c'est-à-dire une version bitmap des données vectorielles de l'occurrence, données non destinées à être considérablement modifiées tout au long de la vie de votre fichier SWF. Ainsi, les occurrences pour lesquelles la mise en cache est activée ne sont pas continuellement redessinées pendant la lecture du fichier SWF, et le rendu de ce dernier est plus rapide.

Remarque : *vous pouvez mettre à jour les données vectorielles au moment de la recréation de la surface. Ainsi, les données vectorielles mises en cache dans la surface ne doivent pas nécessairement être les mêmes pour l'ensemble du fichier SWF.*

Pour qu'un objet d'affichage mette en cache sa représentation sous forme de bitmap, il suffit d'activer sa propriété `cacheAsBitmap` (`true`). Flash Player ou AIR crée un objet de surface pour l'occurrence, correspondant à une image bitmap mise en cache et non à des données vectorielles. Si vous modifiez les limites de l'objet d'affichage, la surface est recréée et non redimensionnée. Les surfaces peuvent s'imbriquer dans d'autres surfaces. La surface copiera l'image bitmap sur sa surface parent. Pour plus d'informations, voir « Activation de la mise en cache sous forme de bitmap » à la page 192.

Les propriétés `opaqueBackground` et `scrollRect` de la classe DisplayObject sont liées à la mise en cache sous forme de bitmap par le biais de la propriété `cacheAsBitmap`. Bien que ces trois propriétés soient indépendantes l'une de l'autre, `opaqueBackground` et `scrollRect` ne sont utiles que lorsqu'un objet est mis en cache sous forme de bitmap. Les avantages des propriétés `opaqueBackground` et `scrollRect` en termes de performances ne sont visibles que si `cacheAsBitmap` est définie sur `true`. Pour plus d'informations sur le défilement du contenu des objets d'affichage, voir « Défilement horizontal ou vertical des objets d'affichage » à la page 185. Pour plus d'informations sur la définition d'un arrière-plan opaque, voir « Définition d'un arrière-plan opaque » à la page 192.

Pour plus d'informations sur le masquage du canal alpha, qui demande que vous définissiez la propriété `cacheAsBitmap` sur `true`, voir « Masquage des objets d'affichage » à la page 197.

Quand activer la mise en cache

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'activation de la mise en cache d'un objet d'affichage crée une surface dont les avantages sont multiples, par exemple pour accélérer le rendu des animations vectorielles complexes. Lorsque vous souhaitez activer la mise en cache, plusieurs scénarios sont disponibles. Il pourrait sembler avantageux de toujours activer la mise en cache pour améliorer les performances de votre fichier SWF. Cependant, dans certains cas, cette opération n'a aucun effet bénéfique et risque même parfois de ralentir le fichier. Cette section présente des cas où la mise en cache s'avère bénéfique, et d'autres où il est préférable d'utiliser les objets de façon normale.

Les performances générales des données mises en cache dépendent de la complexité des données vectorielles de vos occurrences, de la quantité de modifications et de la définition, ou non, de la propriété `opaqueBackground`. Si vous modifiez de petites zones, la différence entre l'utilisation d'une surface et celle de données vectorielles sera négligeable. Vous pouvez dans ce cas tester les deux scénarios avant de déployer votre application.

Quand utiliser la mise en cache sous forme de bitmap

Ce qui suit est une série de scénarios dans lesquels vous pouvez voir les bénéfices significatifs qui résultent de la mise en cache sous forme de bitmap.

- Image complexe d'arrière-plan : application qui contient une image d'arrière-plan complexe de données vectorielles (peut-être une image à laquelle vous avez appliqué la commande de traçage de bitmap ou illustration créée dans Adobe Illustrator®). Vous pouvez animer les caractères sur l'arrière-plan, ce qui ralentit l'animation parce que l'arrière-plan a besoin de continuellement régénérer les données vectorielles. Pour améliorer les performances, vous pouvez définir la propriété `opaqueBackground` de l'objet d'affichage d'arrière-plan sur `true`. L'arrière-plan est rendu en tant que bitmap et peut être redessiné rapidement pour que l'animation soit lue beaucoup plus vite.
- Champ de texte de défilement : application qui affiche une grande quantité de texte dans un champ de texte de défilement. Vous pouvez placer le champ de texte dans un objet d'affichage que vous définissez comme déroulant à l'aide de bornes de déroulement (propriété `scrollRect`). Ceci permet un déroulement de pixels rapide pour l'occurrence indiquée. Quand un utilisateur déroule l'occurrence d'objet d'affichage, Flash Player ou AIR fait défiler les pixels déroulés vers le haut et génère la zone nouvellement exposée au lieu de régénérer tout le champ de texte.

- Système de fenêtres : application comportant un système complexe de chevauchement de fenêtres. Chaque fenêtre peut être ouverte ou fermée (par exemple, les fenêtres du navigateur Web). Si vous marquez chaque fenêtre en tant que surface (en définissant la propriété `cacheAsBitmap` sur `true`), chaque fenêtre est isolée et mise en cache. Les utilisateurs peuvent faire glisser les fenêtres de manière à ce qu'elles se chevauchent. Chaque fenêtre n'a pas besoin de régénérer le contenu vectoriel.
- Masquage du canal alpha : si vous utilisez le masquage du canal alpha, vous devez définir la propriété `cacheAsBitmap` sur `true`. Pour plus d'informations, voir « Masquage des objets d'affichage » à la page 197.

Activer la mise en cache sous forme de bitmap dans tous ces scénarios améliore la réactivité et l'interactivité de l'application en optimisant les graphiques vectoriels.

Par ailleurs, lorsque vous appliquez un filtre à un objet d'affichage, `cacheAsBitmap` est automatiquement définie sur `true`, même si vous l'avez explicitement définie sur `false`. Si vous supprimez tous les filtres appliqués à l'objet d'affichage, la propriété `cacheAsBitmap` retrouve la valeur précédemment définie.

Quand éviter d'utiliser la mise en cache sous forme de bitmap

L'utilisation à mauvais escient de cette fonction risque d'affecter les performances du fichier SWF. Avant d'utiliser la mise en cache sous forme de bitmap, tenez compte des considérations suivantes :

- N'abusez pas des surfaces (objets d'affichage avec mise en cache activée). Chaque surface utilise plus de mémoire qu'un objet d'affichage standard, ce qui signifie que vous ne devez activer les surfaces que lorsqu'il est nécessaire d'améliorer les performances de rendu.

  Un bitmap caché utilise beaucoup plus de mémoire qu'un objet d'affichage standard. Par exemple, si la taille d'une occurrence de Sprite sur la scène correspond à 250 pixels sur 250 pixels, elle est susceptible d'utiliser en cache 250 Ko au lieu d'1 Ko pour une occurrence de Sprite standard (non en cache).
- Evitez de zoomer dans les surfaces cachées. Si vous abusez de la mise en cache sous forme de bitmap, une grande quantité de mémoire sera occupée (voir la puce précédente), surtout si vous zoomez sur le contenu.
- Utilisez des surfaces essentiellement non statiques (sans animation) pour les occurrences d'objets d'affichage. Vous pouvez faire glisser ou déplacer l'occurrence, mais son contenu ne doit pas être animé ni subir de nombreuses modifications (les animations ou les contenus qui changent sont plus fréquents lorsqu'une occurrence de MovieClip contient une animation ou une occurrence de Video). Par exemple, si vous faites pivoter ou transformez une occurrence, la différence entre la surface et les données vectorielles rend le traitement difficile et affecte le fichier SWF.
- Panacher des surfaces avec des données vectorielles accroît la charge de traitement de Flash Player et AIR (et quelquefois de l'ordinateur). Dans la mesure du possible, regroupez les surfaces, en particulier lorsque vous créez des applications à fenêtres.
- Ne mettez pas en cache les objets dont les graphiques sont fréquemment modifiés. A chaque fois que vous exécutez une mise à l'échelle, inclinaison ou rotation de l'objet d'affichage, modifiez l'alpha ou la transformation de couleur, déplacez des objets d'affichage enfant ou tracez à l'aide de la propriété graphique, la mémoire cache des bitmaps est régénérée. Si cette opération se produit à chaque image, le moteur d'exécution doit tracer l'objet dans une image bitmap, puis copier celle-ci sur la scène, d'où une charge de travail accrue par rapport au simple tracé de l'objet non mis en cache sur la scène. L'impact sur les performances de la mise en cache par rapport à la fréquence de mise à jour varie selon la complexité et la taille de l'objet d'affichage et ne peut être déterminé qu'en testant le contenu concerné.

Activation de la mise en cache sous forme de bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour activer la mise en cache d'un objet d'affichage sous forme de bitmap, définissez sa propriété `cacheAsBitmap` sur `true` :

```
mySprite.cacheAsBitmap = true;
```

Après avoir défini la propriété `cacheAsBitmap` sur `true`, vous remarquerez que l'objet d'affichage accroche automatiquement les pixels sur les coordonnées entières. Lorsque vous testez le fichier SWF, vous devriez remarquer que le rendu d'une animation associée à une image vectorielle complexe est bien plus rapide.

Une surface (bitmap en cache) n'est pas créée même quand `cacheAsBitmap` est défini sur `true` s'il se produit l'un ou l'autre des événements suivants :

- Le bitmap fait plus de 2 880 pixels en hauteur ou en largeur.
- Il est impossible d'allouer de la mémoire pour l'image bitmap.

Matrices de transformation des images bitmap mises en cache

Adobe AIR 2.0 et les versions ultérieures (profil mobile)

Dans les applications AIR pour périphériques mobiles, vous devriez définir la propriété `cacheAsBitmapMatrix` à chaque fois que vous définissez la propriété `cacheAsBitmap`. Définir cette propriété permet d'appliquer un large éventail de transformations à l'objet d'affichage sans déclencher un nouveau rendu.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

Lorsque vous définissez cette propriété, vous pouvez appliquer la transformation complémentaire suivante à l'objet d'affichage sans mettre à nouveau l'objet en cache :

- Déplacement ou translation sans accrochage de pixel
- Rotation
- Mise à l'échelle
- Inclinaison
- Modification de l'alpha (transparence comprise entre 0 et 100 %)

Ces transformations sont appliquées directement à l'image bitmap mise en cache.

Définition d'un arrière-plan opaque

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez définir un arrière-plan opaque pour un objet d'affichage. Par exemple, si l'arrière-plan de votre fichier SWF contient une illustration vectorielle complexe, vous pouvez définir la propriété `opaqueBackground` sur une couleur donnée (en général la même couleur que la scène). La couleur est stipulée sous forme de nombre (généralement une valeur hexadécimale). L'arrière-plan est alors considéré comme un bitmap, ce qui permet d'optimiser les performances.

Lorsque vous définissez `cacheAsBitmap` sur `true` et la propriété `opaqueBackground` sur une couleur donnée, la propriété `opaqueBackground` assure l'opacité du bitmap interne et un rendu plus rapide. Si vous ne définissez pas `cacheAsBitmap` sur `true`, la propriété `opaqueBackground` ajoute une forme carrée vectorielle opaque à l'arrière-plan de l'objet d'affichage. Elle ne crée pas automatiquement un bitmap.

L'exemple suivant décrit comment définir l'arrière-plan d'un objet d'affichage pour optimiser les performances :

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

Dans ce cas, la couleur d'arrière-plan de l'objet Shape appelé `myShape` est définie sur rouge (`0xFF0000`). Si l'on suppose que l'occurrence de Shape contient un triangle vert, sur une scène dotée d'un fond blanc le résultat sera un triangle vert sur le fond rouge défini par le cadre de sélection de l'occurrence de Shape (le rectangle qui entoure entièrement Shape).

Ce code serait bien entendu plus logique s'il était utilisé avec une scène dotée d'un arrière-plan rouge uni. Si l'arrière-plan était d'une autre couleur, celle-ci remplacerait le rouge. Par exemple, dans un fichier SWF doté d'un arrière-plan blanc, la propriété `opaqueBackground` serait probablement définie sur `0xFFFFFF`, soit un blanc pur.

Application de modes de fondu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les modes de fondu supposent d'associer les couleurs d'une image (l'image de base) à celles d'une autre image (l'image mélangée) afin de produire une troisième image, qui est celle qui sera en fait affichée à l'écran. Chaque valeur de pixels d'une image est traitée avec la valeur de pixels correspondante de l'autre image afin d'obtenir un résultat présentant une valeur de pixels de position identique.

Tous les objets d'affichage possèdent une propriété `blendMode` qui peut être définie sur l'un des modes de fondu ci-dessous. Les valeurs ci-dessous sont des constantes définies dans la classe BlendMode. Vous pouvez également utiliser les valeurs String (entre parenthèses) correspondant aux valeurs réelles des constantes.

- `BlendMode.ADD` (`"add"`) : s'utilise couramment pour créer un effet de dissolution animée entre deux images en éclaircissant progressivement leurs couleurs.
- `BlendMode.ALPHA` (`"alpha"`) : fréquemment utilisé pour appliquer la transparence de l'avant-plan à l'arrière-plan. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.DARKEN` (`"darken"`) : s'utilise couramment pour superposer un type. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.DIFFERENCE` (`"difference"`) : s'utilise couramment pour créer des couleurs plus vives.
- `BlendMode.ERASE` (`"erase"`) : s'utilise couramment pour découper (effacer) une partie de l'arrière-plan à l'aide de l'alpha d'avant-plan. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.HARDLIGHT` (`"hardlight"`) : s'utilise couramment pour créer des effets d'ombrage. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.INVERT` (`"invert"`) : permet d'inverser l'arrière-plan.
- `BlendMode.LAYER` (`"layer"`) : permet d'imposer la création d'un tampon provisoire en vue de la précomposition d'un objet d'affichage spécifique. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.LIGHTEN` (`"lighten"`) : s'utilise couramment pour superposer un type. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.MULTIPLY` (`"multiply"`) : s'utilise couramment pour créer des ombres et des effets de profondeur.

- `BlendMode.NORMAL` (`"normal"`) : permet aux valeurs des pixels de l'image mélangée de supplanter celles de l'image de base.
- `BlendMode.OVERLAY` (`"overlay"`) : s'utilise couramment pour créer des effets d'ombrage. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.SCREEN` (`"screen"`) : s'utilise couramment pour créer des mises en surbrillance et des halos.
- `BlendMode.SHADER` (`"shader"`) : permet d'indiquer l'utilisation d'un shader de Pixel Bender pour créer un effet de mélange personnalisé. Pour plus d'informations sur l'utilisation des shaders, voir « Utilisation des shaders de Pixel Bender » à la page 310. (Pas de prise en charge en cas de rendu par processeur graphique.)
- `BlendMode.SUBTRACT` (`"subtract"`) : s'utilise couramment pour créer un effet de dissolution animée entre deux images en assombrissant progressivement leurs couleurs.

Réglage des couleurs DisplayObject

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser les méthodes de la classe ColorTransform (flash.geom.ColorTransform) pour régler la couleur d'un objet d'affichage. Chaque objet d'affichage possède une propriété `transform`, qui est une occurrence de la classe Transform et contient des informations relatives aux diverses transformations appliquées à l'objet d'affichage (rotation, changements d'échelle ou de position, etc.). Outre ces informations sur les transformations géométriques, la classe Transform comprend également une propriété `colorTransform`, qui est une occurrence de la classe ColorTransform et permet de régler les couleurs de l'objet d'affichage Pour accéder aux informations de transformation d'un objet d'affichage, utilisez le code suivant :

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

Après avoir créé une occurrence de ColorTransform, vous pouvez lire les valeurs de ses propriétés pour connaître les transformations de couleur qui lui ont déjà été appliquées, et vous pouvez modifier ces valeurs pour changer les couleurs de l'objet. Pour mettre à jour l'objet d'affichage après toute modification, vous devez réaffecter l'occurrence de ColorTransform à la propriété `transform.colorTransform`.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

Définition des valeurs de couleur à l'aide du code

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `color` de la classe ColorTransform permet d'affecter une valeur de couleur rouge, vert, bleu (RVB) déterminée à l'objet d'affichage. L'exemple suivant utilise la propriété `color` pour changer en bleu la couleur de l'objet d'affichage `square` lorsque l'utilisateur clique sur le bouton `blueBtn` :

```
// square is a display object on the Stage.
// blueBtn, redBtn, greenBtn, and blackBtn are buttons on the Stage.

import flash.events.MouseEvent;
import flash.geom.ColorTransform;

// Get access to the ColorTransform instance associated with square.
var colorInfo:ColorTransform = square.transform.colorTransform;

// This function is called when blueBtn is clicked.
function makeBlue(event:MouseEvent):void
{
    // Set the color of the ColorTransform object.
    colorInfo.color = 0x003399;
    // apply the change to the display object
    square.transform.colorTransform = colorInfo;
}

blueBtn.addEventListener(MouseEvent.CLICK, makeBlue);
```

Notez que si vous changez la couleur d'un objet d'affichage à l'aide de la propriété `color`, c'est la couleur de l'objet entier qui est modifiée, même s'il comportait plusieurs couleurs à l'origine. Par exemple, si un objet d'affichage contient un cercle vert en arrière-plan d'un texte noir, le choix du rouge pour la propriété `color` de l'occurrence de ColorTransform associée à cet objet transformera intégralement l'objet en rouge, cercle et texte compris (le texte ne sera donc plus lisible sur le fond de la même couleur).

Modification des effets de couleur et de luminosité à l'aide du code

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Supposons qu'un objet d'affichage comporte plusieurs couleurs (par exemple une photo numérique) et que vous n'ayez pas l'intention de modifier la couleur de l'ensemble de l'objet, mais uniquement celle d'un de ses éléments sur la base des couleurs existantes. Dans ce scénario, la classe ColorTransform comprend une série de propriétés de multiplication et de dominante permettant d'effectuer ce type de réglage. Les propriétés de multiplication, appelées `redMultiplier`, `greenMultiplier`, `blueMultiplier` et `alphaMultiplier`, fonctionnent comme des filtres photographiques de couleur (ou des lunettes à verres de couleur) et amplifient ou diminuent certaines couleurs de l'objet d'affichage. Les propriétés de dominante (`redOffset`, `greenOffset`, `blueOffset` et `alphaOffset`) permettent d'ajouter à l'objet une quantité supplémentaire d'une certaine couleur, ou d'indiquer la valeur minimale que peut avoir une couleur particulière.

Ces propriétés de multiplication et de dominante sont identiques aux réglages de couleurs avancés relatifs aux symboles de clips dans l'outil de programmation Flash lorsque vous sélectionnez Paramètres avancés dans le menu déroulant Couleur de l'Inspecteur des propriétés.

Le code suivant charge une image JPEG et lui applique une transformation de couleur, qui ajuste les canaux rouge et vert lorsque le pointeur de la classe se déplace le long des axes x et y. Dans ce cas précis, comme aucune valeur de dominante n'est spécifiée, les valeurs de chaque canal colorimétrique seront un pourcentage de la valeur de couleur originale de l'image, c'est-à-dire que la valeur maximale de rouge ou de vert d'un pixel donné sera la quantité originale de vert ou de rouge de ce pixel.

```
import flash.display.Loader;
import flash.events.MouseEvent;
import flash.geom.Transform;
import flash.geom.ColorTransform;
import flash.net.URLRequest;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// This function is called when the mouse moves over the loaded image.
function adjustColor(event:MouseEvent):void
{
    // Access the ColorTransform object for the Loader (containing the image)
    var colorTransformer:ColorTransform = loader.transform.colorTransform;

    // Set the red and green multipliers according to the mouse position.
    // The red value ranges from 0% (no red) when the cursor is at the left
    // to 100% red (normal image appearance) when the cursor is at the right.
    // The same applies to the green channel, except it's controlled by the
    // position of the mouse in the y axis.
    colorTransformer.redMultiplier = (loader.mouseX / loader.width) * 1;
    colorTransformer.greenMultiplier = (loader.mouseY / loader.height) * 1;

    // Apply the changes to the display object.
    loader.transform.colorTransform = colorTransformer;
}

loader.addEventListener(MouseEvent.MOUSE_MOVE, adjustColor);
```

Rotation des objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `rotation` permet de faire pivoter les objets d'affichage. Vous pouvez lire cette valeur pour savoir si un objet a été soumis à une rotation. Vous pouvez également la définir sur un nombre (exprimé en degrés) représentant le montant de rotation à appliquer à l'objet. Par exemple, cette ligne de code fait pivoter l'objet `square` de 45 degrés (un huitième de tour complet) :

```
square.rotation = 45;
```

Vous pouvez également faire pivoter un objet d'affichage par le biais d'une matrice de transformation, décrite dans « Utilisation de la géométrie » à la page 218.

Application d'effets de fondu à des objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez contrôler la transparence d'un objet d'affichage de sorte à le rendre partiellement (ou totalement) transparent, ou modifier la transparence pour créer une impression d'apparition ou de disparition en fondu de l'objet. La propriété `alpha` de la classe DisplayObject définit la transparence (ou plus précisément l'opacité) d'un objet d'affichage. La propriété `alpha` peut être définie sur n'importe quelle valeur comprise entre 0 et 1, sachant que 0 correspond à une transparence totale et 1 à une opacité totale. Par exemple, le code suivant rend l'objet `myBall` transparent à 50 % lors d'un clic de souris :

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

Vous pouvez également modifier la transparence d'un objet d'affichage en utilisant les réglages de couleur proposés par la classe ColorTransform. Pour plus d'informations, voir « Réglage des couleurs DisplayObject » à la page 194.

Masquage des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser un objet d'affichage comme masque pour créer un « trou » laissant apparaître le contenu d'un autre objet.

Définition d'un masque

Pour indiquer qu'un objet d'affichage sera le masque d'un autre objet d'affichage, définissez l'objet masque comme propriété `mask` de l'objet à masquer :

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

L'objet d'affichage masqué est révélé sous toutes les zones opaques (non transparentes) de l'objet d'affichage servant de masque. Par exemple, le code suivant crée une occurrence de Shape qui contient un carré rouge de 100 pixels sur 100 et une occurrence de Sprite contenant un cercle bleu d'un rayon de 25 pixels. Lorsque l'utilisateur clique sur le cercle, il devient le masque du carré, de sorte que l'unique partie du carré affichée est la section couverte par la partie pleine du cercle. En d'autres termes, seul un cercle rouge est visible.

```
// This code assumes it's being run within a display object container
// such as a MovieClip or Sprite instance.

import flash.display.Shape;

// Draw a square and add it to the display list.
var square:Shape = new Shape();
square.graphics.lineStyle(1, 0x000000);
square.graphics.beginFill(0xff0000);
square.graphics.drawRect(0, 0, 100, 100);
square.graphics.endFill();
this.addChild(square);

// Draw a circle and add it to the display list.
var circle:Sprite = new Sprite();
circle.graphics.lineStyle(1, 0x000000);
circle.graphics.beginFill(0x0000ff);
circle.graphics.drawCircle(25, 25, 25);
circle.graphics.endFill();
this.addChild(circle);

function maskSquare(event:MouseEvent):void
{
    square.mask = circle;
    circle.removeEventListener(MouseEvent.CLICK, maskSquare);
}

circle.addEventListener(MouseEvent.CLICK, maskSquare);
```

L'objet d'affichage qui sert de masque peut être glissé, animé, redimensionné dynamiquement et peut utiliser plusieurs formes au sein d'un masque unique. Il n'est pas nécessaire que l'objet qui fait office de masque soit ajouté à la liste d'affichage. Toutefois, pour que l'objet servant de masque soit mis à l'échelle lors de la mise à l'échelle de la scène ou pour activer l'interaction utilisateur avec le masque (opération de type glisser et redimensionner contrôlée par l'utilisateur, par exemple), vous devez l'ajouter à la liste d'affichage. L'indice de profondeur (z-index, pour l'ordre de superposition) des objets d'affichage n'a pas d'importance, dès lors que l'objet masque est ajouté à la liste d'affichage (l'objet servant de masque ne s'affiche pas à l'écran, sauf en tant que masque). Si l'objet servant de masque est une occurrence de MovieClip dotée de plusieurs images, il lit toutes les images de son scénario, comme il le ferait s'il ne faisait pas office de masque. Pour supprimer un masque, définissez la propriété `mask` sur `null` :

```
// remove the mask from mySprite
mySprite.mask = null;
```

Il est impossible d'utiliser un masque pour en masquer un autre. Il est impossible de définir la propriété `_alpha` d'un objet d'affichage servant de masque. Seuls les remplissages sont utilisés dans un objet d'affichage employé comme masque ; les traits sont ignorés.

AIR 2

Si un objet d'affichage masqué est mis en cache en définissant les propriétés `cacheAsBitmap` et `cacheAsBitmapMatrix`, le masque doit correspondre à un enfant de l'objet d'affichage masqué. De même, si l'objet d'affichage masqué est un descendant d'un conteneur d'objet d'affichage mis en cache, le masque et l'objet d'affichage doivent tous deux être des descendants du conteneur. Si l'objet masqué est un descendant de plusieurs conteneurs d'objet d'affichage mis en cache, le masque doit être un descendant du conteneur mis en cache le plus proche de l'objet masqué dans la liste d'affichage.

A propos du masquage des polices de périphérique

Vous pouvez utiliser un objet d'affichage pour masquer le texte défini dans une police de périphérique. Dans ce cas, le cadre de sélection rectangulaire du masque est utilisé comme forme de masquage. Ainsi, si vous créez un masque objet d'affichage non rectangulaire pour un texte de police de périphérique, le masque qui apparaît dans le fichier SWF prend la forme du cadre de sélection rectangulaire du masque, et non celle du masque en tant que tel.

Masquage du canal alpha

Le masquage du canal alpha est pris en charge si le masque et les objets d'affichage masqués utilisent la mise en cache sous forme de bitmap, comme illustré ci-après :

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

Une application du masquage du canal alpha consiste par exemple à appliquer un filtre à l'objet masque indépendamment d'un filtre appliqué à l'objet d'affichage masqué.

Dans l'exemple suivant, un fichier d'image externe est chargé sur la scène. Cette image (ou, plus précisément, l'occurrence de Loader dans laquelle elle est chargée) correspondra à l'objet d'affichage masqué. Un ovale dégradé (centre noir uni dont les bords deviennent progressivement transparents) est dessiné sur l'image. Il s'agit-là du masque alpha. La mise en cache sous forme de bitmap est activée pour les deux objets d'affichage. L'ovale est défini en tant que masque de l'image, puis peut être déplacé.

```
// This code assumes it's being run within a display object container
// such as a MovieClip or Sprite instance.

import flash.display.GradientType;
import flash.display.Loader;
import flash.display.Sprite;
import flash.geom.Matrix;
import flash.net.URLRequest;

// Load an image and add it to the display list.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// Create a Sprite.
var oval:Sprite = new Sprite();
// Draw a gradient oval.
var colors:Array = [0x000000, 0x000000];
var alphas:Array = [1, 0];
var ratios:Array = [0, 255];
```

```
var matrix:Matrix = new Matrix();
matrix.createGradientBox(200, 100, 0, -100, -50);
oval.graphics.beginGradientFill(GradientType.RADIAL,
                                colors,
                                alphas,
                                ratios,
                                matrix);
oval.graphics.drawEllipse(-100, -50, 200, 100);
oval.graphics.endFill();
// add the Sprite to the display list
this.addChild(oval);

// Set cacheAsBitmap = true for both display objects.
loader.cacheAsBitmap = true;
oval.cacheAsBitmap = true;
// Set the oval as the mask for the loader (and its child, the loaded image)
loader.mask = oval;

// Make the oval draggable.
oval.startDrag(true);
```

Animation des objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'animation consiste à faire bouger un élément ou à le faire progressivement évoluer. Les animations par script représentent un élément fondamental des jeux vidéo, et elles sont aussi fréquemment utilisées pour obtenir un résultat plus séduisant et ajouter des interactions à d'autres applications.

Une animation par script est régie par un principe de base : il doit se produire une évolution et cette dernière doit être divisée en incréments, au fil du temps. Il est facile de répéter une opération en ActionScript, à l'aide d'une simple boucle. Toutefois, une boucle exécute toutes ses itérations avant de mettre à jour l'affichage. Pour créer une animation par script, vous devez écrire un code ActionScript qui exécute plusieurs fois une action au fil du temps et met également à jour l'écran à chaque exécution.

Supposons par exemple que vous souhaitez créer une animation simple, telle qu'une balle qui traverse l'écran. ActionScript comprend un mécanisme simple qui permet de suivre le passage du temps et de mettre à jour l'écran en conséquence. En d'autres termes, vous pourriez écrire un code qui déplace la balle d'un petit incrément à chaque fois jusqu'à ce qu'elle atteigne sa destination. Après chaque déplacement, l'écran serait mis à jour, afin que l'utilisateur puisse visualiser le mouvement à l'écran.

D'un point de vue pratique, il est logique de synchroniser l'animation par script avec la cadence du fichier SWF (en d'autres termes, de modifier une animation à chaque fois qu'une nouvelle image s'affiche ou devrait s'afficher), puisque cela permet de définir la fréquence des mises à jour de l'écran par Flash Player ou AIR. Chaque objet d'affichage possède un événement `enterFrame` qui est diffusé en fonction de la cadence d'affichage du fichier SWF (un événement par image). La plupart des développeurs qui créent une animation par script utilisent l'événement `enterFrame` pour générer des actions répétées au fil du temps. Vous pourriez écrire du code qui écoute l'événement `enterFrame`, déplace la balle animée d'un incrément déterminé à chaque image et, lorsque l'écran est mis à jour (à chaque image), la balle est redessinée à sa nouvelle position, créant ainsi un mouvement.

Remarque : *une autre technique pour exécuter une action de manière répétitive dans le temps consiste à utiliser la classe Timer. Une occurrence de Timer déclenche une notification d'événement après un délai horaire donné. Il est donc possible d'écrire du code effectuant une animation sur la base de l'événement timer de la classe Timer, en définissant un intervalle très court (une fraction de seconde). Pour plus d'informations sur l'utilisation de la classe Timer, voir «* *Contrôle des intervalles temporels » à la page 4.*

Dans l'exemple suivant, une occurrence circulaire de Sprite, appelée `circle`, est créée sur la scène. Lorsque l'utilisateur clique sur le cercle, une séquence animée par script débute et entraîne un fondu de `circle` (sa propriété `alpha` diminue) jusqu'à ce qu'il soit complètement transparent :

```
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;

// draw a circle and add it to the display list
var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x990000);
circle.graphics.drawCircle(50, 50, 50);
circle.graphics.endFill();
addChild(circle);

// When this animation starts, this function is called every frame.
// The change made by this function (updated to the screen every
// frame) is what causes the animation to occur.
function fadeCircle(event:Event):void
{
    circle.alpha -= .05;

    if (circle.alpha <= 0)
    {
        circle.removeEventListener(Event.ENTER_FRAME, fadeCircle);
    }
}

function startAnimation(event:MouseEvent):void
{
    circle.addEventListener(Event.ENTER_FRAME, fadeCircle);
}

circle.addEventListener(MouseEvent.CLICK, startAnimation);
```

Lorsque l'utilisateur clique sur le cercle, la fonction `fadeCircle()` est enregistrée en tant qu'écouteur de l'événement `enterFrame`. En d'autres termes, elle commence à être appelée une fois par image. Cette fonction provoque un fondu de l'objet `circle` en changeant sa propriété `alpha`, si bien qu'à chaque nouvelle image la valeur de la propriété `alpha` du cercle décroît de 0,05 (soit 5 %) et l'écran est actualisé. Au fil du temps, lorsque la valeur `alpha` correspond à 0 (auquel cas `circle` est complètement transparent), la fonction `fadeCircle()` est supprimée des écouteurs d'événement et l'animation s'arrête.

Le même code permet, par exemple, de créer un mouvement animé au lieu d'un fondu. En substituant une autre propriété à `alpha` dans la fonction qui écoute l'événement `enterFrame`, cette propriété est animée. Par exemple, remplacer la ligne

```
    circle.alpha -= .05;
```

par la ligne

```
    circle.x += 5;
```

anime la propriété `x`. Le cercle se déplace alors vers la droite de la scène. La condition qui arrête l'animation peut être modifiée de sorte à arrêter l'animation (en d'autres termes, annuler l'enregistrement de l'écouteur `enterFrame`) lorsque la coordonnée x appropriée est atteinte.

Orientation de la scène

AIR 2.0 et les versions ultérieures

Les périphériques mobiles réorientent généralement l'interface utilisateur de sorte à assurer un affichage à la verticale lorsque l'utilisateur fait pivoter le périphérique. Si vous activez l'orientation automatique dans votre application, le périphérique conserve l'orientation adéquate de l'écran, mais il vous incombe de vérifier que le contenu s'affiche correctement lorsque le format de la scène change. Si vous désactivez l'orientation automatique, l'écran du périphérique reste fixe, à moins que vous ne modifiiez l'orientation manuellement.

Les applications AIR s'exécutent sur divers systèmes d'exploitation et périphériques mobiles. Le comportement d'orientation sous-jacent risque de varier selon le système d'exploitation, voire selon le périphérique sur un même système d'exploitation. Une stratégie de création simple adaptée à tous les périphériques et systèmes d'exploitation consiste à activer l'orientation automatique et à écouter les événements `resize` de l'objet Stage pour déterminer lorsqu'il est nécessaire d'actualiser la mise en forme de l'application.

Si l'application prend en charge le format portrait uniquement ou le format paysage uniquement, vous pouvez également désactiver l'orientation automatique et définir le format géré dans le descripteur d'application AIR. Cette stratégie assure un comportement cohérent et sélectionne l'orientation la « plus » adaptée au format sélectionné. Par exemple, si vous activez le format paysage, l'orientation choisie convient aux périphériques dotés de claviers coulissants (en mode paysage).

Identification du format et de l'orientation actuels de la scène

L'orientation est indiquée relativement à la position normale du périphérique. Sur la plupart des périphériques, il existe une position verticale clairement identifiable. Cette position est considérée comme l'orientation *par défaut*. Les trois autres orientations possibles sont les suivantes : *rotated left*, *rotated right* et *upside down*. La classe StageOrientation intègre les constantes de type chaîne à utiliser lors de la définition ou de la comparaison de valeurs d'orientation.

La classe Stage définit deux propriétés qui indiquent l'orientation, à savoir :

- Stage.deviceOrientation : indique l'orientation physique du périphérique par rapport à la position par défaut.

  Remarque : *l'orientation du périphérique n'est pas toujours disponible si l'application vient de démarrer ou si le périphérique est posé à plat. Dans ces cas de figure, l'orientation indiquée du périphérique correspond à unknown.*

- Stage.orientation : indique l'orientation de la scène par rapport à la position par défaut. Si vous avez activé l'orientation automatique, la scène pivote dans la direction opposée à celle du périphérique pour demeurer verticale. Les positions droite et gauche indiquées par la propriété `orientation` représentent donc l'opposé des positions indiquées par la propriété `deviceOrientation`. Ainsi, si `deviceRotation` indique *rotated right*, `orientation` indique *rotated left*.

Pour identifier le format de la scène, il suffit de comparer la largeur et la hauteur actuelles de la scène :

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ?
StageAspectRatio.LANDSCAPE : StageAspectRatio.PORTRAIT;
```

Orientation automatique

Si l'orientation automatique est activée et que l'utilisateur fait pivoter le périphérique, le système d'exploitation réoriente l'interface utilisateur entière, y compris la barre des tâches système et l'application. Le format de la scène passe du mode portrait au mode paysage, et vice versa. La modification du format entraîne également la modification des dimensions de la scène.

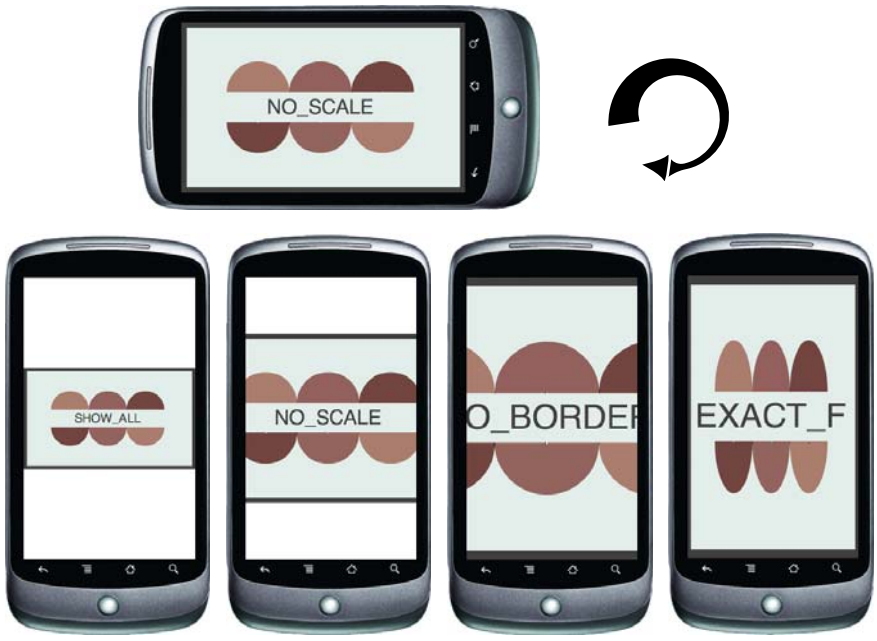
Pour activer ou désactiver l'orientation automatique lors de l'exécution, définissez la propriété `autoOrients` de l'objet Stage sur `true` ou `false`. Vous pouvez définir la valeur initiale de cette propriété dans le descripteur d'application AIR avec l'élément `<autoOrients>`. (Notez que dans les versions d'AIR antérieures à 2.6, `autoOrients` est une propriété disponible en lecture seule, que vous ne pouvez définir que dans le descripteur d'application.)

Si vous spécifiez le format paysage ou portrait et activez l'orientation automatique, AIR définit l'orientation automatique au format spécifié.

Modification des dimensions de la scène

En cas de modification des dimensions de la scène, le contenu de cette dernière est mis à l'échelle et repositionné comme stipulé par les propriétés `scaleMode` et `align` de l'objet Stage. Dans la plupart des cas, il n'est pas recommandé de se fier au comportement automatique défini par les paramètres `scaleMode` de l'objet Stage. Pour prendre en charge plusieurs formats, il est préférable de modifier la mise en forme des graphiques et des composants ou de rafraîchir ces derniers. (Une logique de mise en forme souple présente également l'avantage d'assurer un meilleur fonctionnement de l'application sur des périphériques aux tailles d'écran et formats distincts.)

L'illustration suivante présente les effets des différents paramètres `scaleMode` lors de la rotation d'un périphérique mobile standard :



Rotation d'un format paysage vers un format portrait

Cette illustration décrit le comportement du périphérique lorsque l'utilisateur passe du mode paysage au mode portrait avec différents modes de mise à l'échelle. Le passage du mode portrait au mode paysage produit des effets similaires.

Evénements de changement d'orientation

L'objet Stage distribue deux types d'événements, dont vous disposez pour détecter les changements d'orientation et réagir à ces derniers. Les événements `resize` et `orientationChange` de la scène sont tous deux distribués si l'orientation automatique est activée.

Privilégiez l'événement *resize* si vous vous fiez à l'orientation automatique pour assurer une position verticale à l'affichage. Lorsque la scène distribue un événement `resize`, changez la disposition de votre contenu ou redessinez-le si nécessaire. L'événement `resize` n'est distribué que si le mode de mise à l'échelle de la scène est défini sur `noScale`.

L'événement `orientationChange` permet également de détecter les changements d'orientation. L'événement `orientationChange` n'est distribué que si l'orientation automatique est activée.

Remarque : *sur certaines plates-formes mobiles, la scène distribue un événement `orientationChanging`, que vous pouvez annuler, avant de distribuer les événements resize ou orientationChange. Etant donné que cet événement n'est pas pris en charge sur toutes les plates-formes, évitez de trop vous y fier.*

Orientation manuelle

AIR 2.6 et les versions ultérieures

Vous pouvez contrôler l'orientation de la scène via la méthode `setOrientation()` ou `setAspectRatio()` de l'objet Stage.

Définition de l'orientation de la scène

Vous pouvez définir l'orientation de la scène à l'exécution à l'aide de la méthode `setOrientation()` de l'objet Stage. Utilisez les constantes de chaîne définies par la classe StageOrientation pour spécifier l'orientation de votre choix :

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

Tous les périphériques et systèmes d'exploitation ne prennent pas en charge toutes les orientations possibles. Par exemple, Android 2.2 ne prend pas en charge la sélection par programmation de l'orientation vers la gauche sur les périphériques portrait standard et ne prend pas du tout en charge l'orientation à l'envers. La propriété `supportedOrientations` de la scène fournit une liste des orientations pouvant être transmises à la méthode `setOrientation()` :

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

Définition du format de la scène

Si le format de la scène vous importe, vous pouvez le définir sur portrait ou paysage. Vous pouvez définir le format soit dans le descripteur de l'application AIR, soit à l'exécution à l'aide de la méthode `setAspectRatio()` de l'objet Stage :

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

Le moteur d'exécution choisit l'une des deux orientations possibles pour le format spécifié. Il est possible que ce choix ne corresponde pas à l'orientation du périphérique. Par exemple, l'orientation par défaut est préférée à l'orientation à l'envers (AIR 3.2 et les versions antérieures) et l'orientation convenant au clavier coulissant est préférée à l'orientation opposée.

(AIR 3.3 et versions ultérieures) A partir d'AIR 3.3 (SWF version 16), vous pouvez également utiliser la constante `StageAspectRatio.ANY`. Si vous définissez `Stage.autoOrients` sur `true` et appelez `setAspectRatio(StageAspectRatio.ANY)`, votre application a la capacité de réorienter toutes les orientations (paysage vers la gauche, paysage vers la droite, portait et portrait à l'envers). Autre nouveauté dans AIR 3.3 : le format est persistent et toute rotation du périphérique est soumise à l'orientation spécifiée.

Exemple : définition de l'orientation de la scène de façon à ce qu'elle corresponde à l'orientation du périphérique
L'exemple suivant illustre une fonction qui met à jour l'orientation de la scène afin qu'elle corresponde à celle du périphérique. La propriété `deviceOrientation` de la scène indique l'orientation physique du périphérique, même si l'orientation automatique est désactivée.

```
function refreshOrientation( theStage:Stage ):void
{
    switch ( theStage.deviceOrientation )
    {
        case StageOrientation.DEFAULT:
            theStage.setOrientation( StageOrientation.DEFAULT );
            break;
        case StageOrientation.ROTATED_RIGHT:
            theStage.setOrientation( StageOrientation.ROTATED_LEFT );
            break;
        case StageOrientation.ROTATED_LEFT:
            theStage.setOrientation( StageOrientation.ROTATED_RIGHT );
            break;
        case StageOrientation.UPSIDE_DOWN:
            theStage.setOrientation( StageOrientation.UPSIDE_DOWN );
            break;
        default:
            //No change
    }
}
```

La modification de l'orientation est asynchrone. Vous pouvez écouter l'événement `orientationChange` distribué par la scène pour savoir quand la modification est terminée. Si une orientation n'est pas prise en charge sur un périphérique, l'appel de `setOrientation()` échoue sans distribuer d'erreur.

Chargement dynamique du contenu d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les éléments d'affichage externes suivants peuvent être chargés dans une application ActionScript 3.0 :

- Fichier SWF programmé dans ActionScript 3.0 : ce fichier peut correspondre à Sprite, MovieClip ou toute classe qui étend Sprite. Dans les applications AIR sous iOS, seuls les fichiers SWF qui ne contiennent pas de code d'octet ActionScript peuvent être chargés. En d'autres termes, les fichiers SWF qui contiennent des données incorporées, telles qu'images et son, peuvent être chargées, mais pas les fichiers SWF contenant du code exécutable.
- Fichier d'image : tels que les fichiers JPG, PNG et GIF.
- Fichier AVM1 SWF : fichier SWF écrit en ActionScript 1.0 ou 2.0. (non pris en charge sur les applications mobiles AIR)

Vous chargez ces ressources par le biais de la classe Loader.

Chargement d'objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les objets Loader permettent de charger des fichiers SWF et des fichiers graphiques dans une application. La classe Loader est une sous-classe de la classe DisplayObjectContainer. La liste d'affichage d'un objet Loader ne comporte qu'un seul objet d'affichage enfant : l'objet d'affichage qui représente le fichier SWF ou graphique qu'il charge. Lorsque vous ajoutez un objet Loader à la liste d'affichage, comme dans le code ci-dessous, vous ajoutez également l'objet d'affichage enfant chargé à la liste d'affichage, une fois le chargement effectué :

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

Lorsque le fichier SWF ou l'image sont chargés, vous pouvez transférer l'objet d'affichage chargé dans un autre conteneur d'objets d'affichage, tel que l'objet `container` de la classe DisplayObjectContainer dans l'exemple illustré :

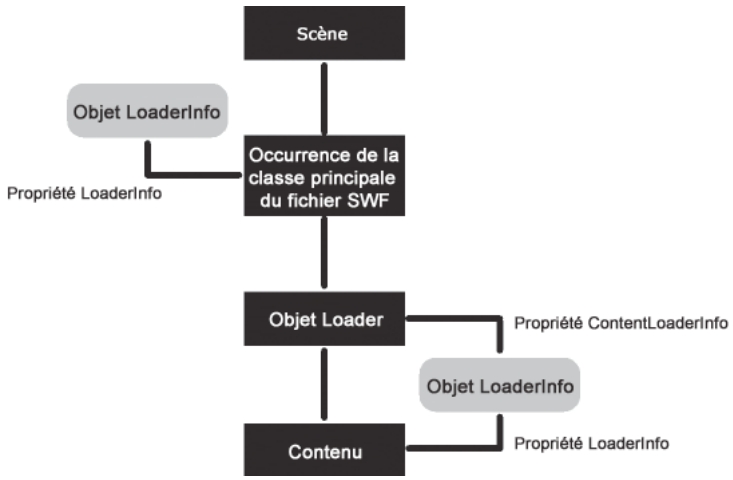
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

Surveillance de la progression du chargement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque le chargement du fichier débute, un objet LoaderInfo est créé. Un objet LoaderInfo fournit diverses informations sur le chargement : progression, adresses URL du chargeur et du chargé, nombre d'octets total de l'objet multimédia et dimensions nominales (hauteur et largeur) de celui-ci. Par ailleurs, un objet LoaderInfo distribue les événements qui permettent de suivre la progression du chargement.

Le diagramme suivant présente les diverses utilisations de l'objet LoaderInfo, pour l'occurrence de la classe principale du fichier SWF, pour un objet Loader et pour un objet chargé par ce dernier :



Vous pouvez accéder à l'objet LoaderInfo en tant que propriété de l'objet Loader et de l'objet d'affichage chargé. Dès que le chargement débute, vous pouvez accéder à l'objet LoaderInfo par le biais de la propriété `contentLoaderInfo` de l'objet Loader. Lorsque le chargement de l'objet d'affichage est terminé, vous pouvez également accéder à l'objet LoaderInfo en tant que propriété de cet objet chargé par le biais de la propriété `loaderInfo` de l'objet d'affichage. La propriété `loaderInfo` de l'objet d'affichage chargé se réfère au même objet LoaderInfo que la propriété `contentLoaderInfo` de l'objet Loader. En d'autres termes, un objet LoaderInfo est partagé entre un objet chargé et l'objet Loader qui l'a chargé (entre le chargeur et le chargé).

Pour accéder aux propriétés du contenu chargé, il est recommandé d'ajouter un écouteur d'événement à l'objet LoaderInfo, comme indiqué dans le code suivant :

```
import flash.display.Loader;
import flash.display.Sprite;
import flash.events.Event;

var ldr:Loader = new Loader();
var urlReq:URLRequest = new URLRequest("Circle.swf");
ldr.load(urlReq);
ldr.contentLoaderInfo.addEventListener(Event.COMPLETE, loaded);
addChild(ldr);

function loaded(event:Event):void
{
    var content:Sprite = event.target.content;
    content.scaleX = 2;
}
```

Pour plus d'informations, voir « Gestion des événements » à la page 129.

Définition du contexte de chargement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous chargez un fichier externe dans Flash Player ou AIR par le biais de la méthode `load()` ou `loadBytes()` de la classe Loader, vous pouvez indiquer le paramètre `context`. Ce paramètre est un objet LoaderContext.

La classe LoaderContext comporte trois propriétés qui permettent de définir le contexte d'utilisation du contenu chargé :

- `checkPolicyFile` : utilisez cette propriété uniquement pour le chargement d'un fichier image (pas pour un fichier SWF). Si vous définissez cette propriété sur `true`, l'objet Loader vérifie si le serveur d'origine héberge un fichier de régulation (voir « Contrôles de site Web (fichiers de régulation) » à la page 1095). Cette opération n'est requise que si le contenu émane de domaines autres que celui du fichier SWF qui contient l'objet Loader. Si le serveur accorde une autorisation au domaine de Loader, le code ActionScript extrait des fichiers SWF du domaine de Loader peut accéder aux données de l'image chargée. En d'autres termes, vous pouvez utiliser la commande `BitmapData.draw()` pour accéder aux données de l'image chargées.

  Notez qu'un fichier SWF extrait d'un autre domaine que celui de l'objet Loader peut appeler `Security.allowDomain()` pour autoriser un domaine déterminé.

- `securityDomain` : utilisez cette propriété uniquement pour le chargement d'un fichier SWF (pas pour une image). Cette propriété peut être appelée pour un fichier SWF provenant d'un autre domaine que celui du fichier qui contient l'objet Loader. Lorsque vous indiquez cette option, Flash Player vérifie l'existence d'un fichier de régulation et, s'il existe, les fichiers SWF des domaines autorisés dans ce fichier peuvent utiliser des opérations de programmation croisée avec le contenu du fichier SWF chargé. Vous pouvez stipuler `flash.system.SecurityDomain.currentDomain` en tant que paramètre.

- `applicationDomain` : utilisez cette propriété uniquement lors du chargement d'un fichier SWF écrit dans ActionScript 3.0 (et non une image ou un fichier SWF écrit dans ActionScript 1.0 ou 2.0). Lorsque vous chargez un fichier, vous devez indiquer que le fichier doit être inclus dans le même domaine d'application que l'objet Loader en attribuant au paramètre `applicationDomain` la valeur `flash.system.ApplicationDomain.currentDomain`. Si vous placez le fichier SWF chargé dans le même domaine d'application, vous pourrez accéder directement à ses classes, ce qui s'avère utile si vous chargez un fichier SWF contenant des média intégrés, auxquels vous pouvez accéder via les noms de classes associés. Pour plus d'informations, voir « Utilisation de domaines d'application » à la page 152.

Exemple de vérification d'un fichier de régulation lors du chargement d'une image bitmap provenant d'un autre domaine :

```
var context:LoaderContext = new LoaderContext();
context.checkPolicyFile = true;
var urlReq:URLRequest = new URLRequest("http://www.[your_domain_here].com/photo11.jpg");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

Exemple de vérification d'un fichier de régulation lors du chargement d'un fichier SWF à partir d'un autre domaine, dans le but de placer ce fichier dans la même Sandbox de sécurité que l'objet Loader. Par ailleurs, le code ajoute les classes du fichier SWF chargé dans le même domaine d'application que celui de l'objet Loader :

```
var context:LoaderContext = new LoaderContext();
context.securityDomain = SecurityDomain.currentDomain;
context.applicationDomain = ApplicationDomain.currentDomain;
var urlReq:URLRequest = new URLRequest("http://www.[your_domain_here].com/library.swf");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

Pour plus d'informations, voir la classe LoaderContext dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Chargement de fichiers SWF dans AIR pour iOS

Adobe AIR 3.6 et ultérieur, iOS uniquement

Il existe des restrictions concernant le chargement et la compilation de code à l'exécution sur les périphériques iOS. En raison de ces restrictions, vous constaterez certaines différences nécessaires dans la tâche de chargement des fichiers SWF externes dans votre application :

- Tous les fichiers contenant du code ActionScript doivent être inclus dans le package d'application. Aucun fichier SWF contenant du code ne peut être chargé depuis une source externe (via un réseau, par exemple). Lors de la création du package de l'application, le code ActionScript des fichiers SWF du package d'application est compilé en code natif pour périphériques iOS.
- Il est impossible de charger, décharger, puis recharger un fichier SWF. Si vous tentez de le faire, une erreur se produit.
- Le comportement en cas de chargement en mémoire, puis de déchargement, est le même qu'avec les systèmes d'exploitation d'ordinateur de bureau. Si vous chargez un fichier SWF, puis que vous le déchargez, tous les éléments visuels contenus dans le SWF sont déchargés de la mémoire. Toutefois, les références de classe à une classe ActionScript dans le fichier SWF chargé restent en mémoire et sont accessibles par le code ActionScript.
- Tous les fichiers SWF doivent être chargés dans le même domaine d'application que le fichier SWF principal. Ceci n'est pas le comportement par défaut. C'est pourquoi, pour chaque SWF chargé, vous devez créer un objet LoaderContext spécifiant le domaine d'application principal et transmettre cet objet LoaderContext à l'appel de méthode Loader.load(). Si vous tentez de charger un fichier SWF dans un domaine d'application autre que le domaine d'application SWF principal, une erreur se produit. Cela est vrai même si le fichier SWF chargé ne contient que des éléments visuels sans code ActionScript.

  L'exemple suivant montre le code à utiliser pour charger le SWF depuis le package d'application dans le domaine d'application du fichier SWF principal :

```
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("swfs/SecondarySwf.swf");
var loaderContext:LoaderContext = new LoaderContext(false, ApplicationDomain.currentDomain,
null);
loader.load(url, loaderContext);
```

Un fichier SWF contenant uniquement des actifs sans code peut être chargé depuis le package d'application ou via réseau. Dans les deux cas, le fichier SWF doit toujours être chargé dans le domaine d'application principal.

Pour les versions d'AIR antérieures à 3.6, le code est retiré de tous les fichiers SWF, à l'exception du fichier d'application SWF principal, au cours du processus de compilation. Les fichiers SWF ne contenant que des éléments visuels peuvent être inclus dans le package d'application et chargés à l'exécution, mais pas le code. Si vous tentez de charger un fichier SWF contenant du code ActionScript, une erreur se produit. Cette erreur cause l'apparition d'un message d'erreur « ActionScript non compilé » dans l'application.

Voir aussi

Packaging and loading multiple SWFs in AIR apps on iOS (disponible en anglais uniquement).

Utilisation des classes ProLoader et ProLoaderInfo

Flash Player 9 et les versions ultérieures et Adobe AIR 1.0 et les versions ultérieures requièrent Flash Professional CS5.5

Pour faciliter le préchargement de la bibliothèque RSL (Remote Shared Library), Flash Professional CS5.5 intègre à présent les classes fl.display.ProLoader et fl.display.ProLoaderInfo. Ces classes reflètent les classes flash.display.Loader et flash.display.LoaderInfo, mais assurent un chargement plus cohérent.

La classe ProLoader permet en particulier de charger les fichiers SWF qui font appel à Text Layout Framework (TLF) lors d'un préchargement RSL. A l'exécution, les fichiers SWF qui préchargent d'autres fichiers SWF ou des fichiers SWZ, tels que TLF, nécessitent un fichier d'enveloppe SWF à usage interne uniquement. Cette couche complémentaire de complexité imposée par le fichier d'enveloppe SWF entraîne parfois un comportement inattendu. La classe ProLoader résout ce problème en permettant de charger les fichiers comme des fichiers SWF standard. La solution adoptée par la classe ProLoader est transparente du point de vue de l'utilisateur et ne requiert pas de traitement particulier dans ActionScript. La classe ProLoader charge par ailleurs correctement les contenus SWF standard.

Dans Flash Professional CS 5.5 et ultérieur, vous pouvez remplacer la classe Loader par la classe ProLoader dans tous les cas de figure en toute sécurité. Exportez ensuite l'application vers Flash Player 10.2 ou ultérieur, afin que la classe ProLoader puisse accéder à la fonctionnalité ActionScript requise. Vous pouvez également faire appel à la classe ProLoader si vous ciblez des versions antérieures de Flash Player qui prennent en charge ActionScript 3.0. Toutefois, seul Flash Player 10.2 ou ultérieur exploite pleinement les fonctions de Proloader. Utilisez toujours la classe ProLoader si vous faites appel à TLF dans Flash Professional CS5.5 ou ultérieur. L'utilisation de ProLoader est superflue dans les environnements autres que Flash Professional.

Important : *pour les fichiers SWF publiés dans Flash Professional CS5.5 et ultérieur, il est toujours possible d'utiliser les classes fl.display.ProLoader et fl.display.ProLoaderInfo au lieu des classes flash.display.Loader et flash.display.LoaderInfo.*

Problèmes résolus par la classe ProLoader

La classe ProLoader résout les problèmes que la classe Loader existante, de par sa conception, ne prenait pas en charge. Ces problèmes résultent du préchargement RSL de bibliothèques TLF. Ils concernent spécifiquement les fichiers SWF qui chargent d'autres fichiers SWF par le biais d'un objet Loader. Les problèmes résolus sont les suivants :

- **L'utilisation de scripts entre le fichier de chargement et le fichier chargé entraîne des résultats inattendus.** La classe ProLoader définit automatiquement le fichier SWF de chargement en tant que parent du fichier SWF chargé. De ce fait, les communications émanant du fichier SWF de chargement ciblent directement le fichier SWF chargé.
- **L'application SWF doit gérer activement le processus de chargement.** Cette opération requiert la mise en œuvre d'événements supplémentaires tels que `added`, `removed`, `addedToStage` et `removedFromStage`. Si l'application cible Flash Player 10.2 ou ultérieur, la classe ProLoader permet d'éviter cette tâche supplémentaire.

Mise à jour du code en vue d'utiliser ProLoader au lieu de Loader

Etant donné que la classe ProLoader reflète la classe Loader, substituer une classe à l'autre dans le code ne présente aucune difficulté. L'exemple suivant illustre la mise à jour du code existant en vue d'utiliser la nouvelle classe :

```
import flash.display.Loader;
import flash.events.Event;
var l:Loader = new Loader();

addChild(l);
l.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete);
l.load("my.swf");
function loadComplete(e:Event) {
    trace('load complete!');
}
```

Il est possible de mettre à jour ce code pour utiliser la classe ProLoader, comme suit :

```
import fl.display.ProLoader;
import flash.events.Event;
var l:ProLoader = new ProLoader();

addChild(l);
l.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete);
l.load("my.swf");
function loadComplete(e:Event) {
    trace('load complete!');
}
```

Exemple d'objet d'affichage : SpriteArranger

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple d'application SpriteArranger est basé sur l'exemple d'application Geometric Shapes (voir *Formation à ActionScript 3.0*).

L'exemple SpriteArranger illustre divers concepts de gestion des objets d'affichage :

- Extension des classes d'objet d'affichage
- Ajout d'objets à la liste d'affichage
- Superposition des objets d'affichage et utilisation des conteneurs d'objets d'affichage
- Réponse aux événements d'objet d'affichage
- Utilisation des propriétés et méthodes des objets d'affichage

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application SpriteArranger résident dans le dossier Examples/SpriteArranger. L'application se compose des fichiers suivants :

Fichier	Description
SpriteArranger.mxml ou SpriteArranger.fla	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/SpriteArranger/CircleSprite.as	Classe définissant un type d'objet Sprite qui assure le rendu d'un cercle à l'écran.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	Classe définissant le rectangle de la zone de dessin, soit un conteneur d'objets d'affichage comportant des objets GeometricSprite.
com/example/programmingas3/SpriteArranger/SquareSprite.as	Classe définissant un type d'objet Sprite qui assure le rendu d'un carré à l'écran.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	Classe définissant un type d'objet Sprite qui assure le rendu d'un triangle à l'écran.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	Classe qui étend l'objet Sprite, utilisé pour définir une forme à l'écran. CircleSprite, SquareSprite et TriangleSprite étendent chacun cette classe.
com/example/programmingas3/geometricshapes/IGeometricShape.as	Interface de base qui définit les méthodes à implémenter par toutes les classes de formes géométriques.
com/example/programmingas3/geometricshapes/IPolygon.as	Interface qui définit les méthodes à implémenter par les classes de forme géométrique dotées de plusieurs côtés.
com/example/programmingas3/geometricshapes/RegularPolygon.as	Type de forme géométrique dont les côtés sont de longueur égale et positionnés symétriquement autour du centre de la forme.
com/example/programmingas3/geometricshapes/Circle.as	Type de forme géométrique qui définit un cercle.
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	Sous-classe de RegularPolygon qui définit un triangle équilatéral.
com/example/programmingas3/geometricshapes/Square.as	Sous-classe de RegularPolygon qui définit un carré.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	Classe qui contient une « méthode usine » permettant de créer des formes d'une taille et d'un type de forme donnés.

Définition des classes SpriteArranger

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application SpriteArranger permet à l'utilisateur d'ajouter divers objets d'affichage au rectangle de la zone de dessin à l'écran.

La classe DrawingCanvas définit une zone de dessin, soit un type de conteneur d'objets d'affichage, à laquelle l'utilisateur peut ajouter des formes à l'écran. Ces formes sont des occurrences de l'une des sous-classes de la classe GeometricSprite.

Classe DrawingCanvas

Dans Flex, tous les objets d'affichage enfant ajoutés à un objet Container doivent appartenir à une classe issue de la classe mx.core.UIComponent. Cette application ajoute une occurrence de la classe DrawingCanvas en tant qu'enfant d'un objet mx.containers.VBox, ainsi que le définit le code MXML dans le fichier SpriteArranger.mxml. Cet héritage est défini dans la déclaration de classe DrawingCanvas, comme suit :

```
public class DrawingCanvas extends UIComponent
```

La classe UIComponent héritage des classes DisplayObject, DisplayObjectContainer et Sprite et le code de la classe DrawingCanvas utilise les méthodes et propriétés de ces dernières.

La classe DrawingCanvas étend la classe Sprite et cet héritage est défini dans la déclaration de classe DrawingCanvas, comme suit :

```
public class DrawingCanvas extends Sprite
```

La classe Sprite est une sous-classe des classes DisplayObjectContainer et DisplayObject et la classe DrawingCanvas utilise les méthodes et propriétés de ces dernières.

La méthode constructeur `DrawingCanvas()` définit un objet Rectangle, `bounds`, qui est une propriété utilisée ultérieurement lors du tracé du contour de la zone de dessin. Elle appelle ensuite la méthode `initCanvas()`, comme suit :

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

Comme l'indique l'exemple suivant, la méthode `initCanvas()` définit diverses propriétés de l'objet DrawingCanvas, transmises en tant qu'arguments à la méthode constructeur :

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

La méthode `initCanvas()` appelle ensuite la méthode `drawBounds()` qui trace le rectangle de la zone de dessin à l'aide de la propriété `graphics` de la classe DrawingCanvas. La propriété `graphics` est héritée de la classe Shape.

```
this.graphics.clear();
this.graphics.lineStyle(1.0, this.lineColor, 1.0);
this.graphics.beginFill(this.fillColor, 1.0);
this.graphics.drawRect(bounds.left - 1,
                       bounds.top - 1,
                       bounds.width + 2,
                       bounds.height + 2);
this.graphics.endFill();
```

Les autres méthodes de la classe DrawingCanvas, indiquées ci-après, sont appelées en fonction des interactions de l'utilisateur avec l'application :

- Les méthodes `addShape()` et `describeChildren()`, décrites à la section « Ajout d'objets d'affichage au rectangle de la zone de dessin » à la page 214
- Les méthodes `moveToBack()`, `moveDown()`, `moveToFront()` et `moveUp()`, décrites à la section « Réorganisation de l'ordre de superposition des objets d'affichage » à la page 217
- La méthode `onMouseUp()`, décrite à la section « Cliquer-déplacer un objet d'affichage » à la page 216

Classe GeometricSprite et ses sous-classes

Chaque objet d'affichage susceptible d'être ajouté par l'utilisateur au rectangle de la zone de dessin est une occurrence de l'une des sous-classes suivantes de la classe GeometricSprite :

- CircleSprite
- SquareSprite
- TriangleSprite

La classe GeometricSprite étend la classe flash.display.Sprite :

```
public class GeometricSprite extends Sprite
```

La classe GeometricSprite comprend diverses propriétés communes à tous les objets GeometricSprite. Elles sont définies dans la fonction constructeur en fonction des paramètres transmis à cette dernière. Exemple :

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

La propriété `geometricShape` de la classe GeometricSprite définit une interface IGeometricShape, qui stipule les propriétés mathématiques, mais non visuelles, de la forme. Les classes qui implémentent l'interface IGeometricShape sont définies dans l'exemple d'application GeometricShapes (voir *Formation à ActionScript 3.0*).

La classe GeometricSprite définit la méthode `drawShape()`, qui est affinée dans les définitions de remplacement de chaque sous-classe de GeometricSprite. Pour plus d'informations, voir ci-après « Ajout d'objets d'affichage au rectangle de la zone de dessin ».

La classe GeometricSprite propose également les méthodes suivantes :

- Les méthodes `onMouseDown()` et `onMouseUp()`, décrites à la section « Cliquer-déplacer un objet d'affichage » à la page 216
- Les méthodes `showSelected()` et `hideSelected()`, décrites à la section « Cliquer-déplacer un objet d'affichage » à la page 216

Ajout d'objets d'affichage au rectangle de la zone de dessin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur clique sur le bouton Add Shape, l'application appelle la méthode `addShape()` de la classe DrawingCanvas. Elle crée une occurrence de GeometricSprite en appelant la fonction constructeur appropriée de l'une des sous-classes de GeometricSprite, comme illustré dans l'exemple suivant :

```
public function addShape(shapeName:String, len:Number):void
{
    var newShape:GeometricSprite;
    switch (shapeName)
    {
        case "Triangle":
            newShape = new TriangleSprite(len);
            break;

        case "Square":
            newShape = new SquareSprite(len);
            break;

        case "Circle":
            newShape = new CircleSprite(len);
            break;
    }
    newShape.alpha = 0.8;
    this.addChild(newShape);
}
```

Chaque méthode constructeur appelle la méthode `drawShape()`, qui utilise la propriété `graphics` de la classe (héritée de la classe Sprite) pour dessiner le graphique vectoriel approprié. Par exemple, la méthode `drawShape()` de la classe CircleSprite comprend le code suivant :

```
this.graphics.clear();
this.graphics.lineStyle(1.0, this.lineColor, 1.0);
this.graphics.beginFill(this.fillColor, 1.0);
var radius:Number = this.size / 2;
this.graphics.drawCircle(radius, radius, radius);
```

L'avant-dernière ligne de la fonction `addShape()` définit la propriété `alpha` de l'objet d'affichage (héritée de la classe DisplayObject), de sorte que chaque objet d'affichage ajouté au rectangle de la zone de dessin soit légèrement transparent, permettant ainsi à l'utilisateur de visualiser les objets placés derrière.

La derrière ligne de la méthode `addChild()` ajoute le nouvel objet d'affichage à la liste enfant de l'occurrence de la classe DrawingCanvas, qui figure déjà dans la liste d'affichage. Le nouvel objet d'affichage apparaît alors sur la scène.

L'interface de l'application comprend deux champs de texte, `selectedSpriteTxt` et `outputTxt`. Les propriétés de texte de ces champs sont mises à jour avec des informations relatives aux objets GeometricSprite ajoutés au rectangle de la zone de dessin ou sélectionnés par l'utilisateur. La classe GeometricSprite gère ces tâches de transmission d'informations en annulant la méthode `toString()` comme suit :

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

La propriété `shapeType` est définie sur la valeur appropriée de la méthode constructeur de chaque sous-classe GeometricSprite. La méthode `toString()` pourrait par exemple renvoyer la valeur suivante pour une occurrence de CircleSprite récemment ajoutée à l'occurrence de DrawingCanvas :

```
Circle of size 50 at 0, 0
```

La méthode `describeChildren()` de la classe DrawingCanvas parcourt la liste enfant du rectangle de la zone de dessin en utilisant la propriété `numChildren` (héritée de la classe DisplayObjectContainer) pour définir la limite de la boucle `for`. Elle génère une chaîne qui recense chaque enfant, comme suit :

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

La chaîne résultante permet de définir la propriété `text` du champ de texte `outputTxt`.

Cliquer-déplacer un objet d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur clique sur une occurrence de GeometricSprite, l'application appelle le gestionnaire de l'événement `onMouseDown()`. Comme le montre le code ci-dessous, ce gestionnaire écoute les événements de clic gauche de souris dans la fonction constructeur de la classe GeometricSprite :

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

La méthode `onMouseDown()` appelle ensuite la méthode `showSelected()` de l'objet GeometricSprite. S'il s'agit du premier appel de cette méthode sur l'objet, elle crée un objet Shape appelé `selectionIndicator` et utilise la propriété `graphics` de l'objet Shape pour dessiner un rectangle rouge de mise en valeur, comme suit :

```
this.selectionIndicator = new Shape();
this.selectionIndicator.graphics.lineStyle(1.0, 0xFF0000, 1.0);
this.selectionIndicator.graphics.drawRect(-1, -1, this.size + 1, this.size + 1);
this.addChild(this.selectionIndicator);
```

S'il ne s'agit pas du premier appel de la méthode `onMouseDown()`, celle-ci active simplement la propriété `visible` (héritée de la classe DisplayObject) de la forme `selectionIndicator` :

```
this.selectionIndicator.visible = true;
```

La méthode `hideSelected()` masque la forme `selectionIndicator` de l'objet précédemment sélectionné en définissant sa propriété `visible` sur `false`.

La méthode du gestionnaire d'événement `onMouseDown()` appelle également la méthode `startDrag()` (héritée de la classe Sprite), qui comprend le code suivant :

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

L'utilisateur peut alors déplacer l'objet sélectionné sur la zone de dessin, au sein des limites définies par le rectangle `boundsRect`.

Lorsque l'utilisateur relâche le bouton de la souris, l'événement `mouseUp` est distribué. La méthode constructeur de DrawingCanvas configure l'écouteur d'événement suivant :

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

Cet écouteur d'événement est associé à l'objet DrawingCanvas, plutôt qu'aux objets GeometricSprite individuels. En effet, lorsque l'utilisateur déplace l'objet GeometricSprite, celui-ci risque d'être placé derrière un autre objet d'affichage (un autre objet GeometricSprite) lorsque le bouton de la souris est relâché. L'événement « mouse up » s'appliquerait à l'objet d'affichage en avant-plan, mais non à l'objet d'affichage déplacé par l'utilisateur. Ajouter l'écouteur à l'objet DrawingCanvas assure la gestion de l'événement.

La méthode `onMouseUp()` appelle la méthode `onMouseUp()` de l'objet GeometricSprite, qui appelle alors la méthode `stopDrag()` de l'objet GeometricSprite.

Réorganisation de l'ordre de superposition des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'interface utilisateur de l'application comprend des boutons intitulés Move Back, Move Down, Move Up et Move to Front. Lorsque l'utilisateur clique sur l'un de ces boutons, l'application appelle la méthode correspondante de la classe DrawingCanvas, à savoir : `moveToBack()`, `moveDown()`, `moveUp()` ou `moveToFront()`. Par exemple, la méthode `moveToBack()` comporte le code suivant :

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

Cette méthode utilise la méthode `setChildIndex()` (héritée de la classe DisplayObjectContainer) pour placer l'objet d'affichage à la position d'index 0 de la liste des enfants de l'occurrence de DrawingCanvas (`this`).

Le fonctionnement de la méthode `moveDown()` est similaire, mais elle décrémente la position d'index de l'objet d'affichage d'une unité dans la liste des enfants de l'occurrence de DrawingCanvas :

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

Le fonctionnement des méthodes `moveUp()` et `moveToFront()` est similaire aux méthodes `moveToBack()` et `moveDown()`.

Chapitre 11 : Utilisation de la géométrie

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le package flash.geom contient des classes qui définissent des objets géométriques, tels que des points, des rectangles et des matrices de transformation. Vous utilisez ces classes pour définir les propriétés des objets qui sont utilisés dans d'autres classes.

Voir aussi

flash.geom, package

Principes de base de la géométrie

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le package flash.geom contient des classes qui définissent des objets géométriques, tels que des points, des rectangles et des matrices de transformation. Ces classes ne fournissent pas nécessairement de fonctionnalité par elles-mêmes ; néanmoins, elles sont utilisées pour définir les propriétés des objets utilisés dans d'autres classes.

Toutes les classes de géométrie se basent sur la notion selon laquelle les emplacements à l'écran sont représentés comme un plan en deux dimensions. L'écran est traité comme un graphique plat avec un axe horizontal (x) et un axe vertical (y). Tout emplacement (ou *point*) à l'écran peut être représenté sous la forme d'une paire de valeurs x et y, appelées *coordonnées* de cet emplacement.

Chaque objet d'affichage, y compris la scène, possède son propre *espace de coordonnées*. L'espace de coordonnées constitue le graphe d'un objet et permet de tracer la position des dessins, des objets d'affichage enfant, etc. L'*origine* occupe la position 0, 0 (soit l'intersection des axes x et y) et est placée dans l'angle supérieur gauche de l'objet d'affichage. La position de l'origine est systématiquement respectée pour la scène, mais pas nécessairement pour d'autres objets d'affichage. La taille des valeurs figurant sur l'axe x croît vers la droite et diminue vers la gauche. La coordonnée x des positions figurant sur la gauche de l'origine est négative. Cependant, contrairement aux systèmes de coordonnées classiques, la valeur des coordonnées de l'axe y croît vers le bas de l'écran et diminue vers le haut dans le moteur d'exécution de Flash. La coordonnée y des valeurs situées au-dessus de l'origine est négative. Puisque l'angle supérieur gauche de la scène correspond à l'origine de son espace de coordonnées, la coordonnée x de la plupart des objets de la scène est supérieure à 0 et inférieure à la largeur de la scène. La coordonnée y d'un même objet est supérieure à 0 et inférieure à la hauteur de la scène.

Vous pouvez utiliser des occurrences de la classe Point pour représenter des points individuels dans un espace de coordonnées. Vous pouvez créer une occurrence de Rectangle pour représenter une région rectangulaire dans un espace de coordonnées. Les utilisateurs chevronnés peuvent utiliser une occurrence de Matrix pour appliquer des transformations multiples ou complexes à un objet d'affichage. De nombreuses transformations simples (rotation, position, et changements d'échelle, par exemple) peuvent être appliquées directement à un objet d'affichage à l'aide des propriétés de ce dernier. Pour plus d'informations sur l'application de transformations à l'aide des propriétés d'un objet d'affichage, voir « Manipulation des objets d'affichage » à la page 180.

Concepts importants et terminologie
La liste de référence suivante contient des termes de géométrie importants :

Coordonnées cartésiennes Les coordonnées sont généralement écrites sous la forme d'une paire de nombres (5, 12 ou 17, -23). Les deux nombres sont la coordonnée x et la coordonnée y, respectivement.

Espace de coordonnées Représentation graphique des coordonnées contenues dans un objet d'affichage, par rapport auquel sont positionnés les éléments enfant.

Origine Point d'intersection des axes x et y dans un espace de coordonnées. Ce point a les coordonnées 0, 0.

Point Emplacement unique dans un espace de coordonnées. Dans le système de coordonnées 2D utilisé dans ActionScript, la position sur les axes x et y (en d'autres termes, les coordonnées du point) définit ce dernier.

Point d'alignement Dans un objet d'affichage, origine (coordonnées 0, 0) de son espace de coordonnées.

Mise à l'échelle Taille relative d'un objet par rapport à sa taille d'origine. Mettre un objet à l'échelle consiste à modifier sa taille en l'étirant ou en le rétrécissant.

Translation Conversion des coordonnées d'un point d'un espace de coordonnées à un autre.

Transformation Modification des caractéristiques visuelles d'un graphique (rotation de l'objet, modification de son échelle, désalignement, déformation ou altération de sa couleur).

Axe x Axe horizontal dans le système de coordonnées en 2 dimensions utilisé dans ActionScript.

Axe y Axe vertical dans le système de coordonnées en 2 dimensions utilisé dans ActionScript.

Utilisation des objets Point

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet Point définit une paire de coordonnées cartésiennes. Il représente un emplacement dans un système de coordonnées à deux dimensions, dans lequel *x* est l'axe horizontal et *y* l'axe vertical.

Pour définir un objet Point, vous définissez ses propriétés `x` et `y` comme suit :

```
import flash.geom.*;
var pt1:Point = new Point(10, 20); // x == 10; y == 20
var pt2:Point = new Point();
pt2.x = 10;
pt2.y = 20;
```

Calcul de la distance entre deux points

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la méthode `distance()` de la classe Point pour calculer la distance entre deux points dans un espace de coordonnées. Par exemple, le code suivant calcule la distance entre les points d'alignement de deux objets d'affichage, `circle1` et `circle2`, dans le même conteneur d'objet d'affichage :

```
import flash.geom.*;
var pt1:Point = new Point(circle1.x, circle1.y);
var pt2:Point = new Point(circle2.x, circle2.y);
var distance:Number = Point.distance(pt1, pt2);
```

Translation d'espaces de coordonnées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si deux objets d'affichage résident dans des conteneurs d'objet d'affichage distincts, ils peuvent figurer dans des espaces de coordonnées distincts. Vous pouvez utiliser la méthode `localToGlobal()` de la classe DisplayObject pour traduire les coordonnées dans le même espace de coordonnées (global), celui de la scène. Par exemple, le code suivant calcule la distance entre les points d'alignement de deux objets d'affichage, `circle1` et `circle2`, dans les différents conteneurs d'objet d'affichage :

```
import flash.geom.*;
var pt1:Point = new Point(circle1.x, circle1.y);
pt1 = circle1.localToGlobal(pt1);
var pt2:Point = new Point(circle2.x, circle2.y);
pt2 = circle2.localToGlobal(pt2);
var distance:Number = Point.distance(pt1, pt2);
```

De même, pour calculer la distance du point d'alignement d'un objet d'affichage nommé `target` à partir d'un point spécifique de la scène, utilisez la méthode `localToGlobal()` de la classe DisplayObject :

```
import flash.geom.*;
var stageCenter:Point = new Point();
stageCenter.x = this.stage.stageWidth / 2;
stageCenter.y = this.stage.stageHeight / 2;
var targetCenter:Point = new Point(target.x, target.y);
targetCenter = target.localToGlobal(targetCenter);
var distance:Number = Point.distance(stageCenter, targetCenter);
```

Déplacement d'un objet d'affichage en fonction d'une distance et d'un angle donnés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la méthode `polar()` de la classe Point pour déplacer un objet d'affichage d'une distance et d'un angle spécifiques. Par exemple, le code suivant déplace l'objet `myDisplayObject` en fonction d'une distance de 100 pixels et d'un angle de 60° :

```
import flash.geom.*;
var distance:Number = 100;
var angle:Number = 2 * Math.PI * (90 / 360);
var translatePoint:Point = Point.polar(distance, angle);
myDisplayObject.x += translatePoint.x;
myDisplayObject.y += translatePoint.y;
```

Autres utilisations de la classe Point

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser des objets Point avec les propriétés et les méthodes suivantes :

Classe	Méthodes ou propriétés	Description
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()getObjectsUnderPoint()`	Utilisée pour renvoyer une liste d'objets sous un point dans un conteneur d'objet d'affichage.
BitmapData	`hitTest()`	Utilisée pour définir le pixel dans l'objet BitmapData ainsi que le point pour lequel vous recherchez une zone active.

BitmapData	`applyFilter()`
	`copyChannel()`
	`merge()`
	`paletteMap()`
	`pixelDissolve()`
	`threshold()`
		Utilisée pour indiquer les positions des rectangles qui définissent les opérations.

Matrice	`deltaTransformPoint()` `transformPoint()`	Utilisée pour définir des points auxquels vous souhaitez appliquer une transformation.

Rectangle	`bottomRight`
	`size`
	`topLeft`
		Utilisée pour définir ces propriétés.

Utilisation des objets Rectangle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet Rectangle définit une zone rectangulaire. Un objet Rectangle possède une position, définie par les coordonnées *x* et *y* de son angle supérieur gauche, une propriété `width` et une propriété `height`. Pour définir les propriétés d'un nouvel objet Rectangle, appelez la fonction constructeur `Rectangle()`, comme suit :

```
import flash.geom.Rectangle;
var rx:Number = 0;
var ry:Number = 0;
var rwidth:Number = 100;
var rheight:Number = 50;
var rect1:Rectangle = new Rectangle(rx, ry, rwidth, rheight);
```

Redimensionnement et repositionnement des objets Rectangle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il existe de nombreuses façons de redimensionner et de repositionner des objets Rectangle.

Vous pouvez redimensionner directement l'objet Rectangle en modifiant ses propriétés `x` et `y`. Ce changement n'a aucune incidence sur la largeur ou la hauteur de l'objet Rectangle.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.x = 20;
rect1.y = 30;
trace(rect1); // (x=20, y=30, w=100, h=50)
```

Comme l'illustre le code suivant, l'objet Rectangle est repositionné lorsque vous modifiez la propriété `left` ou `top` correspondante. Les propriétés `x` et `y` de l'objet rectangle correspondent respectivement aux propriétés `left` et `top`. Néanmoins, la position de l'angle inférieur gauche de l'objet Rectangle ne change pas. Par conséquent, il est redimensionné.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.left = 20;
rect1.top = 30;
trace(rect1); // (x=20, y=30, w=80, h=20)
```

De même, comme indiqué dans l'exemple suivant, si vous modifiez la propriété `bottom` ou `right` d'un objet Rectangle, la position de son angle supérieur gauche ne change pas. L'objet Rectangle est redimensionné en conséquence :

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.right = 60;
trect1.bottom = 20;
trace(rect1); // (x=0, y=0, w=60, h=20)
```

Vous pouvez également repositionner un objet Rectangle à l'aide de la méthode `offset()`, comme suit :

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.offset(20, 30);
trace(rect1); // (x=20, y=30, w=100, h=50)
```

La méthode `offsetPt()` fonctionne de la même façon, sauf qu'elle prend un objet Point comme paramètre, plutôt que les valeurs de décalage *x* et *y*.

Vous pouvez également redimensionner un objet Rectangle à l'aide de la méthode `inflate()`, qui inclut deux paramètres, `dx` et `dy`. Le paramètre `dx` représente le déplacement à partir du centre des côtés droit et gauche de l'objet Rectangle, exprimé en pixels. Le paramètre `dy` représente le déplacement à partir du centre du haut et du bas de l'objet Rectangle, exprimé en pixels.

```
import flash.geom.Rectangle;
var x1:Number = 0;
var y1:Number = 0;
var width1:Number = 100;
var height1:Number = 50;
var rect1:Rectangle = new Rectangle(x1, y1, width1, height1);
trace(rect1) // (x=0, y=0, w=100, h=50)
rect1.inflate(6,4);
trace(rect1); // (x=-6, y=-4, w=112, h=58)
```

La méthode `inflatePt()` fonctionne de la même façon, sauf qu'elle prend un objet Point comme paramètre, plutôt que les valeurs de décalage `dx` et `dy`.

Recherche d'unions et d'intersections d'objets Rectangle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous utilisez la méthode `union()` pour rechercher la région rectangulaire formée par les limites de deux rectangles :

```
import flash.display.*;
import flash.geom.Rectangle;
var rect1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect1); // (x=0, y=0, w=100, h=100)
var rect2:Rectangle = new Rectangle(120, 60, 100, 100);
trace(rect2); // (x=120, y=60, w=100, h=100)
trace(rect1.union(rect2)); // (x=0, y=0, w=220, h=160)
```

Vous utilisez la méthode `intersection()` pour rechercher la région rectangulaire formée par la région commune de deux rectangles :

```
import flash.display.*;
import flash.geom.Rectangle;
var rect1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(rect1); // (x=0, y=0, w=100, h=100)
var rect2:Rectangle = new Rectangle(80, 60, 100, 100);
trace(rect2); // (x=120, y=60, w=100, h=100)
trace(rect1.intersection(rect2)); // (x=80, y=60, w=20, h=40)
```

Vous utilisez la méthode `intersects()` pour savoir si deux rectangles se recouvrent. Vous pouvez également utiliser la méthode `intersects()` pour savoir si un objet d'affichage est dans une certaine région de la scène. Dans l'exemple de code suivant, supposez que l'espace de coordonnées du conteneur d'objet d'affichage contenant l'objet `circle` soit identique à celui de la scène. L'exemple indique comment utiliser la méthode `intersects()` pour déterminer si un objet d'affichage, `circle`, recoupe des régions spécifiées de la scène, définies par les objets Rectangle `target1` et `target2` :

```
import flash.display.*;
import flash.geom.Rectangle;
var circle:Shape = new Shape();
circle.graphics.lineStyle(2, 0xFF0000);
circle.graphics.drawCircle(250, 250, 100);
addChild(circle);
var circleBounds:Rectangle = circle.getBounds(stage);
var target1:Rectangle = new Rectangle(0, 0, 100, 100);
trace(circleBounds.intersects(target1)); // false
var target2:Rectangle = new Rectangle(0, 0, 300, 300);
trace(circleBounds.intersects(target2)); // true
```

De même, vous pouvez utiliser la méthode `intersects()` pour savoir si les cadres de délimitation de deux objets d'affichage se chevauchent. Utilisez la méthode `getRect()` de la classe DisplayObject pour inclure un espace supplémentaire que les traits d'un objet d'affichage ajoutent à une région de sélection.

Autres utilisations des objets Rectangle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les objets Rectangle sont utilisés dans les propriétés et méthodes suivantes :

Classe	Méthodes ou propriétés	Description
BitmapData	`applyFilter()`, `colorTransform()`, `copyChannel()`, `copyPixels()`, `draw()`, `drawWithQuality()`, `encode()`, `fillRect()`, `generateFilterRect()`, `getColorBoundsRect()`, `getPixels()`, `merge()`, `paletteMap()`, `pixelDissolve()`, `setPixels()` et `threshold()`	Utilisée comme type de certains paramètres pour définir une région de l'objet BitmapData.
DisplayObject	`getBounds()`, `getRect()`, `scrollRect`, `scale9Grid`	Utilisée comme type de données pour la propriété ou le type de données renvoyé.
PrintJob	`addPage()`	Utilisée pour définir le paramètre `printArea`.
Sprite	`startDrag()`	Utilisée pour définir le paramètre `bounds`.
TextField	`getCharBoundaries()`	Utilisée comme type de valeur renvoyé.
Transform	`pixelBounds`	Utilisée comme type de données.

Utilisation des objets Matrix

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Matrix représente une matrice de transformation qui détermine le mappage des points d'un espace de coordonnées à l'autre. Pour appliquer diverses transformations graphiques à un objet d'affichage, vous pouvez définir les propriétés d'un objet Matrix, puis appliquer cet objet à la propriété `matrix` d'un objet Transform que vous appliquez ensuite comme propriété `transform` de l'objet d'affichage. Ces fonctions de transformation incluent la translation (repositionnement de *x* et *y*), la rotation, le redimensionnement et l'inclinaison.

Même si vous pouvez définir une matrice en ajustant directement les propriétés (`a`, `b`, `c`, `d`, `tx`, `ty`) d'un objet Matrix, il est plus facile d'utiliser la méthode `createBox()`. Cette méthode comporte des paramètres qui vous permettent de définir directement les effets de redimensionnement, de rotation et de translation de la matrice résultante. Par exemple, le code suivant crée un objet Matrix qui redimensionne un objet horizontalement de 2,0, le redimensionne verticalement de 3,0, le fait pivoter de 45°, le déplace (par translation) de 10 pixels vers la droite et de 20 pixels vers le bas :

```
var matrix:Matrix = new Matrix();
var scaleX:Number = 2.0;
var scaleY:Number = 3.0;
var rotation:Number = 2 * Math.PI * (45 / 360);
var tx:Number = 10;
var ty:Number = 20;
matrix.createBox(scaleX, scaleY, rotation, tx, ty);
```

Vous pouvez également ajuster les effets de redimensionnement, de rotation et de translation d'un objet Matrix à l'aide des méthodes `scale()`, `rotate()` et `translate()`. Ces méthodes sont combinées aux valeurs de l'objet Matrix existant. Par exemple, le code suivant définit un objet Matrix qui redimensionne un objet d'un facteur de 4 et le fait pivoter de 60°, car les méthodes `scale()` et `rotate()` sont appelées deux fois :

```
var matrix:Matrix = new Matrix();
var rotation:Number = 2 * Math.PI * (30 / 360); // 30°
var scaleFactor:Number = 2;
matrix.scale(scaleFactor, scaleFactor);
matrix.rotate(rotation);
matrix.scale(scaleX, scaleY);
matrix.rotate(rotation);

myDisplayObject.transform.matrix = matrix;
```

Pour appliquer une transformation par inclinaison à un objet Matrix, ajustez sa propriété `b` ou `c`. Lorsque vous ajustez la propriété `b`, la matrice est inclinée verticalement et lorsque vous ajustez la propriété `c`, elle est inclinée horizontalement. Le code suivant incline l'objet Matrix `myMatrix` verticalement d'un facteur de 2 :

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

Vous pouvez appliquer une transformation Matrix à la propriété `transform` d'un objet d'affichage. Par exemple, le code suivant applique une transformation Matrix à un objet d'affichage nommé `myDisplayObject` :

```
var matrix:Matrix = myDisplayObject.transform.matrix;
var scaleFactor:Number = 2;
var rotation:Number = 2 * Math.PI * (60 / 360); // 60°
matrix.scale(scaleFactor, scaleFactor);
matrix.rotate(rotation);

myDisplayObject.transform.matrix = matrix;
```

La première ligne définit un objet Matrix sur la matrice de transformation existante utilisée par l'objet d'affichage `myDisplayObject` (la propriété `matrix` de la propriété `transformation` de l'objet d'affichage `myDisplayObject`). Les méthodes de la classe Matrix que vous appelez ont ainsi un effet cumulatif sur la position, l'échelle et la rotation actuelles de l'objet d'affichage.

Remarque : *la classe ColorTransform est également comprise dans le package flash.geometry. Cette classe sert à définir la propriété* `colorTransform` *d'un objet Transform. Etant donné qu'elle n'applique aucune transformation géométrique, elle n'est pas traitée ici. Pour plus d'informations, voir la classe ColorTransform dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.*

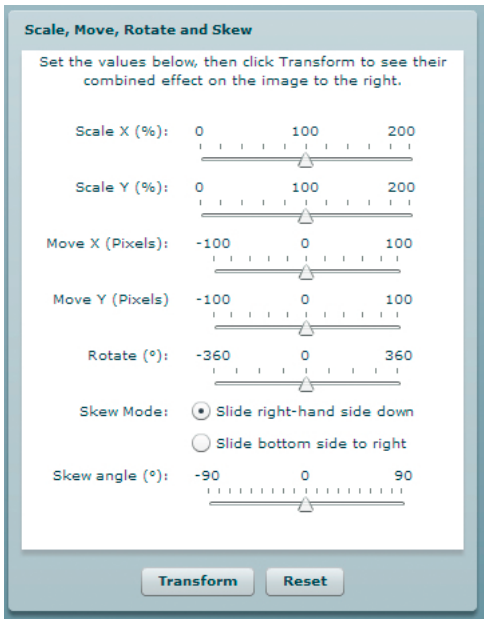
Exemple de géométrie : application d'une transformation de matrice à un objet d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple d'application DisplayObjectTransformer présente de nombreuses fonctions permettant d'utiliser la classe Matrix pour transformer un objet d'affichage, notamment :

- Rotation de l'objet d'affichage
- Redimensionnement de l'objet d'affichage
- Translation (repositionnement) de l'objet d'affichage
- Inclinaison de l'objet d'affichage

L'application fournit une interface permettant d'ajuster les paramètres de la transformation de matrice, comme suit :



Lorsque l'utilisateur clique sur le bouton de transformation, l'application applique la transformation appropriée.

L'objet d'affichage original et l'objet d'affichage pivoté de -45° et redimensionné de 50 %

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application DisplayObjectTransformer se trouvent dans le dossier Samples/DisplayObjectTransformer. L'application se compose des fichiers suivants :

Fichier	Description
DisplayObjectTransformer.mxml ou DisplayObjectTransformer.fla	Fichier d'application principal en FLA pour Flash ou en MXML pour Flex
com/example/programmingas3/geometry/MatrixTransformer.as	Une classe qui contient des méthodes permettant d'appliquer des transformations de matrice.
img/	Un répertoire contenant des exemples de fichiers image utilisés par l'application.

Définition de la classe MatrixTransformer

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe MatrixTransformer comprend des méthodes statiques qui appliquent des transformations géométriques d'objets Matrix.

Méthode transform()

La méthode `transform()` comprend des paramètres associés à chaque élément suivant :

- `sourceMatrix`—La matrice d'entrée, que la méthode transforme
- `xScale` et `yScale`—Le facteur de redimensionnement *x* et *y*
- `dx` et `dy`—Les montants de translation *x* et *y*, en pixels
- `rotation`—Le montant de rotation, en degrés
- `skew`—Le facteur d'inclinaison, en pourcentage
- `skewType`—Le sens d'inclinaison, `"right"` ou `"left"`

La valeur renvoyée est la matrice résultante.

La méthode `transform()` appelle les méthodes statiques suivantes de la classe :

- `skew()`
- `scale()`
- `translate()`
- `rotate()`

Chacune renvoie la matrice source avec la transformation appliquée.

Méthode skew()

La méthode `skew()` incline la matrice en ajustant les propriétés `b` et `c` de la matrice. Un paramètre facultatif, `unit`, détermine les unités utilisées pour définir l'angle d'inclinaison et, le cas échéant, la méthode convertit la valeur `angle` en radians :

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

Un objet Matrix `skewMatrix` est créé et ajusté pour appliquer la transformation par inclinaison. Au départ, il s'agit de la matrice d'identité, comme suit :

```
var skewMatrix:Matrix = new Matrix();
```

Le paramètre `skewSide` détermine le côté auquel l'inclinaison est appliquée. S'il est défini sur `"right"`, le code suivant définit la propriété `b` de la matrice :

```
skewMatrix.b = Math.tan(angle);
```

Autrement, le côté inférieur est incliné en ajustant la propriété `c` de la matrice, comme suit :

```
skewMatrix.c = Math.tan(angle);
```

L'inclinaison résultante est ensuite appliquée à la matrice existante en concaténant les deux matrices, comme indiqué dans l'exemple suivant :

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

Méthode scale()

Comme le montre l'exemple suivant, la méthode `scale()` ajuste d'abord le facteur de redimensionnement s'il est défini sous la forme de pourcentage, puis utilise la méthode `scale()` de l'objet matrix :

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

Méthode translate()

La méthode `translate()` applique simplement les facteurs de translation `dx` et `dy` en appelant la méthode `translate()` de l'objet matrix, comme suit :

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

Méthode rotate()

La méthode `rotate()` convertit le facteur de rotation d'entrée en radians (s'il est fourni en degrés ou dégradés), puis appelle la méthode `rotate()` de l'objet matrix :

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

Appel de la méthode MatrixTransformer.transform() depuis l'application

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application contient une interface utilisateur permettant d'obtenir les paramètres de transformation de l'utilisateur. Elle transmet ensuite ces valeurs, avec la propriété `matrix` de la propriété `transform` de l'objet d'affichage, à la méthode `Matrix.transform()`, comme suit :

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

L'application applique alors la valeur renvoyée à la propriété `matrix` de la propriété `transform` de l'objet d'affichage, déclenchant ainsi la transformation :

```
img.content.transform.matrix = tempMatrix;
```

Chapitre 12 : Utilisation de l'API de dessin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Bien que l'importance des images et graphiques importés soit évidente, il ne faut pas négliger la fonctionnalité connue sous le nom d'API de dessin, qui permet de tracer des lignes et des formes en ActionScript. Vous pouvez ainsi ouvrir une application avec l'équivalent informatique d'une toile vierge et y créer des images. La possibilité de créer des graphiques ouvre de nouvelles possibilités pour vos applications. Les techniques présentées ci-après permettent notamment de créer un programme de dessin, de réaliser des éléments artistiques animés et interactifs ou de générer par programmation vos propres éléments d'interface.

Voir aussi

flash.display.Graphics

Principes de base de l'API de dessin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API de dessin est le nom des fonctionnalités intégrées à ActionScript qui permettent de créer des graphiques vectoriels (lignes, courbes, formes, remplissages et dégradés) et de les afficher à l'aide d'ActionScript. Ces fonctionnalités sont prises en charge par la classe flash.display.Graphics. ActionScript permet de dessiner dans une occurrence d'un objet de type Shape, Sprite ou MovieClip, à l'aide de la propriété `graphics` définie dans chacune de ces classes (la propriété `graphics` de ces classes est en fait une occurrence de la classe Graphics).

Si vous n'avez pas l'habitude de « dessiner » par code, la classe Graphics comprend plusieurs méthodes qui facilitent le tracé de formes courantes (cercles, ellipses, rectangles et rectangles à coins arrondis). Ces tracés peuvent être des lignes vides ou des formes remplies. Si vous avez besoin de fonctionnalités plus sophistiquées, la classe Graphics comporte aussi des méthodes destinées au tracé de lignes et de courbes de Bézier, qui peuvent être utilisées conjointement avec les fonctions trigonométriques de la classe Math pour créer n'importe quelle forme.

Les moteurs d'exécution Flash (tels que Flash Player 10, Adobe AIR 1.5 et les versions ultérieures) prennent en charge une nouvelle API de dessin, qui vous permet de tracer intégralement des formes par programmation à l'aide d'une commande unique. Une fois que vous vous êtes familiarisé avec la classe Graphics et les tâches décrites dans « Bases d'utilisation de l'API de dessin », passez à « Utilisation avancée de l'API de dessin » à la page 243 pour en savoir plus sur ces fonctions.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à l'utilisation de l'API de dessin :

Point d'ancrage L'un des deux points d'extrémité d'une courbe de Bézier.

Point de contrôle Point qui définit la direction et la forme d'une courbe de Bézier. Cette ligne courbe ne touche jamais le point de contrôle, mais elle s'arrondit comme si elle était tracée dans la direction de celui-ci.

Espace de coordonnées Représentation graphique des coordonnées contenues dans un objet d'affichage, par rapport auquel sont positionnés les éléments enfant.

Remplissage Partie intérieure opaque d'une forme constituée par le remplissage d'une ligne ou d'une forme ne possédant pas de ligne de contour.

Dégradé Transition progressive d'une couleur à une ou plusieurs autres couleurs (par opposition à une couleur unie).

Point Emplacement unique dans un espace de coordonnées. Dans le système de coordonnées en 2 dimensions utilisé dans ActionScript, un point est défini par son emplacement le long de l'axe x et de l'axe y (les coordonnées du point).

Courbe de Bézier quadratique Type de courbe défini par une formule mathématique déterminée. Dans ce type de courbe, la forme de la courbe est calculée à partir des positions des points d'ancrage (les points d'extrémité de la courbe) et d'un point de contrôle qui définit la forme et la direction de la courbe.

Mise à l'échelle Taille relative d'un objet par rapport à sa taille d'origine. Mettre un objet à l'échelle consiste à modifier sa taille en l'étirant ou en le rétrécissant.

Trait Ligne de contour d'une forme constituée par le remplissage de cette ligne, ou forme ne possédant pas de remplissage.

Translation Conversion des coordonnées d'un point d'un espace de coordonnées à un autre.

Axe x Axe horizontal dans le système de coordonnées en 2 dimensions utilisé dans ActionScript.

Axe y Axe vertical dans le système de coordonnées en 2 dimensions utilisé dans ActionScript.

Classe Graphics

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque objet Shape, Sprite et MovieClip possède une propriété `graphics` qui est une occurrence de la classe Graphics. La classe Graphics comporte des propriétés et des méthodes permettant de tracer des lignes, des remplissages et des formes. Pour disposer d'un objet d'affichage qui sera uniquement utilisé comme « toile de fond » pour un dessin, utilisez une occurrence de Shape. Une occurrence de Shape est mieux adaptée au dessin que les autres objets, car elle ne comporte pas les fonctionnalités (inutiles dans ce type d'utilisation) des classes Sprite et MovieClip. Par contre, si vous souhaitez créer un objet d'affichage qui servira de support à du contenu graphique mais doit également pouvoir contenir d'autres objets d'affichage, utilisez une occurrence de Sprite. Pour plus d'informations sur le choix des objets d'affichage en fonction de la tâche prévue, voir « Sélection d'une sous-classe de DisplayObject » à la page 179

Dessin de lignes et de courbes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tous les graphiques qu'il est possible de réaliser à l'aide d'une occurrence de la classe Graphics sont des tracés à l'aide de lignes et de courbes. Tous les dessins réalisés en ActionScript doivent donc suivre les mêmes étapes :

- Définition de styles de ligne et de remplissage
- Définition de la position de dessin initiale
- Dessin de lignes, courbes et formes (éventuellement en déplaçant le point de traçage)
- Le cas échéant, création d'un remplissage

Définition de styles de ligne et de remplissage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour utiliser la propriété `graphics` d'une occurrence de Shape, Sprite ou MovieClip, vous devez d'abord définir le style (épaisseur et couleur de la ligne, couleur de remplissage) à utiliser. A l'instar des outils de dessin d'Adobe® Flash® Professional ou de toute autre application de dessin, en ActionScript vous pouvez dessiner avec ou sans trait, et avec ou sans remplissage. Pour indiquer l'apparence du trait, utilisez la méthode `lineStyle()` ou `lineGradientStyle()`. Pour créer une ligne pleine, utilisez la méthode `lineStyle()`. Lors de l'appel de cette méthode, les valeurs les plus courantes à indiquer sont les trois premiers paramètres : épaisseur de ligne, couleur et alpha. Par exemple, la ligne de code suivante indique que la forme `myShape` doit tracer des lignes de deux pixels d'épaisseur, en rouge (0x990000) et avec une opacité de 75 % :

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

La valeur par défaut du paramètre alpha est 1.0 (100 %), vous pouvez donc l'omettre si vous souhaitez tracer une ligne entièrement opaque. La méthode `lineStyle()` gère également deux autres paramètres associés à l'indice de lissage des pixels et au mode de mise à l'échelle. Pour plus d'informations sur ces paramètres, voir la description de la méthode `Graphics.lineStyle()` dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Pour créer une ligne dégradée, utilisez la méthode `lineGradientStyle()`. Pour plus d'informations sur cette méthode, voir « Création de lignes et de remplissages en dégradé » à la page 235.

Pour créer une forme remplie, appelez les méthodes `beginFill()`, `beginGradientFill()`, `beginBitmapFill()` ou `beginShaderFill()` avant de débuter le dessin. La plus basique, `beginFill()`, accepte deux paramètres : la couleur de remplissage et, le cas échéant, la valeur alpha correspondante. Par exemple, pour tracer une forme avec un remplissage vert uni, utilisez le code suivant (on suppose ici que vous dessinez dans un objet nommé `myShape`) :

```
myShape.graphics.beginFill(0x00FF00);
```

L'appel d'une méthode de remplissage annule implicitement le remplissage précédemment défini avant l'implémentation du nouveau. L'appel d'une méthode qui spécifie un style de trait remplace le style de trait précédent, mais ne modifie pas le remplissage précédemment défini, et vice-versa.

Une fois spécifié le style de ligne et de remplissage, l'étape suivante consiste à indiquer le point de départ du dessin. L'occurrence de Graphics possède un point de traçage, tout comme la pointe d'un crayon sur une feuille de papier. Quel que soit l'emplacement du point de traçage, il représente l'origine de l'action de dessin à venir. Initialement, un objet Graphics débute avec son point de traçage aux points 0,0 dans l'espace de coordonnées de l'objet dans lequel il dessine. Pour que le tracé débute en un autre point, appelez la méthode `moveTo()` avant d'appeler une des méthodes de dessin. Cet appel peut être comparé à l'action de lever la pointe du crayon du papier et de l'amener à un nouvel emplacement.

Lorsque le point de traçage est en place, utilisez une série d'appels aux méthodes `lineTo()` (pour tracer des lignes droites) et `curveTo()` (pour tracer des courbes).

Pendant l'opération de dessin, vous pouvez à tout moment appeler la méthode `moveTo()` pour amener le point de traçage à une nouvelle position sans dessiner.

Si vous avez défini une couleur de remplissage, vous pouvez désactiver le remplissage en appelant la méthode `endFill()` pendant l'opération de dessin. Si vous n'avez pas tracé de forme fermée (autrement dit, si lors de l'appel de la méthode `endFill()` le point de traçage ne correspond pas au point de départ de la forme), lorsque vous appelez la méthode `endFill()`, le moteur d'exécution Flash ferme automatiquement la forme en traçant une ligne droite entre le point de traçage actuel et l'emplacement spécifié dans le dernier appel à la méthode `moveTo()`. Si vous avez débuté un remplissage et n'avez pas appelé `endFill()`, tout appel à `beginFill()` (ou à l'une des autres méthodes de remplissage) ferme le remplissage actif et en débute un nouveau.

Dessin de lignes droites

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous appelez la méthode `lineTo()`, l'objet Graphics trace une ligne droite (en utilisant le style de ligne que vous avez spécifié) entre le point de traçage actuel et les coordonnées que vous transmettez en paramètres à cette méthode. Par exemple, cette ligne de code place le point de traçage aux coordonnées 100, 100 puis trace une ligne jusqu'au point 200, 200 :

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

L'exemple suivant trace des triangles rouges et verts d'une hauteur de 100 pixels :

```
var triangleHeight:uint = 100;
var triangle:Shape = new Shape();

// red triangle, starting at point 0, 0
triangle.graphics.beginFill(0xFF0000);
triangle.graphics.moveTo(triangleHeight / 2, 0);
triangle.graphics.lineTo(triangleHeight, triangleHeight);
triangle.graphics.lineTo(0, triangleHeight);
triangle.graphics.lineTo(triangleHeight / 2, 0);

// green triangle, starting at point 200, 0
triangle.graphics.beginFill(0x00FF00);
triangle.graphics.moveTo(200 + triangleHeight / 2, 0);
triangle.graphics.lineTo(200 + triangleHeight, triangleHeight);
triangle.graphics.lineTo(200, triangleHeight);
triangle.graphics.lineTo(200 + triangleHeight / 2, 0);

this.addChild(triangle);
```

Dessin de courbes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `curveTo()` dessine une courbe de Bézier. Elle trace un arc entre deux points (appelés points d'ancrage) courbé en direction d'un troisième point (appelé point de contrôle). L'objet Graphics utilise la position de traçage actuelle comme premier point d'ancrage. Lorsque vous appelez la méthode `curveTo()`, vous transmettez quatre paramètres : les coordonnées x et y du point de contrôle, puis les coordonnées x et y du second point d'ancrage. Par exemple, le code suivant trace une courbe entre le point 100, 100 et le point 200, 200. Le point de contrôle ayant les coordonnées 175, 125, la courbe est orientée vers la droite puis vers le bas :

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

L'exemple suivant trace des objets circulaires rouges et verts avec une largeur et une hauteur de 100 pixels. Notez qu'en raison même de la nature de l'équation de Bézier, ces cercles ne sont pas parfaits :

```
var size:uint = 100;
var roundObject:Shape = new Shape();

// red circular shape
roundObject.graphics.beginFill(0xFF0000);
roundObject.graphics.moveTo(size / 2, 0);
roundObject.graphics.curveTo(size, 0, size, size / 2);
roundObject.graphics.curveTo(size, size, size / 2, size);
roundObject.graphics.curveTo(0, size, 0, size / 2);
roundObject.graphics.curveTo(0, 0, size / 2, 0);

// green circular shape
roundObject.graphics.beginFill(0x00FF00);
roundObject.graphics.moveTo(200 + size / 2, 0);
roundObject.graphics.curveTo(200 + size, 0, 200 + size, size / 2);
roundObject.graphics.curveTo(200 + size, size, 200 + size / 2, size);
roundObject.graphics.curveTo(200, size, 200, size / 2);
roundObject.graphics.curveTo(200, 0, 200 + size / 2, 0);

this.addChild(roundObject);
```

Dessin de formes à l'aide des méthodes intégrées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour vous permettre de tracer plus commodément des formes courantes (cercles, ellipses, rectangles et rectangles à coins arrondis), ActionScript 3.0 comporte des méthodes qui tracent automatiquement ces formes. Ces méthodes sont `drawCircle()`, `drawEllipse()`, `drawRect()` et `drawRoundRect()`, et sont toutes définies dans la classe Graphics. Elles peuvent être utilisées à la place des méthodes `lineTo()` et `curveTo()`. Notez toutefois qu'il est nécessaire de spécifier des styles de ligne et de remplissage avant d'appeler ces méthodes.

L'exemple suivant dessine des carrés bleus, rouges et verts avec une largeur et une hauteur de 100 pixels. Ce code utilise la méthode `drawRect()` et spécifie que l'opacité de la couleur de remplissage est de 50 % (0,5) :

```
var squareSize:uint = 100;
var square:Shape = new Shape();
square.graphics.beginFill(0xFF0000, 0.5);
square.graphics.drawRect(0, 0, squareSize, squareSize);
square.graphics.beginFill(0x00FF00, 0.5);
square.graphics.drawRect(200, 0, squareSize, squareSize);
square.graphics.beginFill(0x0000FF, 0.5);
square.graphics.drawRect(400, 0, squareSize, squareSize);
square.graphics.endFill();
this.addChild(square);
```

Dans un objet Sprite ou MovieClip, tout contenu graphique créé à l'aide de la propriété `graphics` apparaît toujours derrière les objets d'affichage enfant contenus par cet objet. Par ailleurs, le contenu créé avec la propriété `graphics` n'est pas un objet d'affichage séparé. Il n'apparaît donc pas dans la liste des enfants d'un objet Sprite ou MovieClip. Par exemple, l'objet Sprite suivant reçoit l'instruction de tracer un cercle avec sa propriété `graphics`, et un objet TextField figure dans sa liste d'objets d'affichage enfant :

```
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0xFFCC00);
mySprite.graphics.drawCircle(30, 30, 30);
var label:TextField = new TextField();
label.width = 200;
label.text = "They call me mellow yellow...";
label.x = 20;
label.y = 20;
mySprite.addChild(label);
this.addChild(mySprite);
```

Notez que l'objet TextField apparaît au-dessus du cercle tracé avec la propriété graphics.

Création de lignes et de remplissages en dégradé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'objet Graphics permet aussi de tracer des traits et des remplissages avec des dégradés au lieu de couleurs unies. Pour créer un trait dégradé, utilisez la méthode `lineGradientStyle()`. Pour créer un remplissage dégradé, utilisez la méthode `beginGradientFill()`.

Ces deux méthodes reçoivent les mêmes paramètres. Les quatre premiers sont obligatoires : type, couleurs, transparences alpha et rapports. Les quatre suivants sont facultatifs mais peuvent être utiles pour plus de personnalisation.

- Le premier paramètre spécifie le type de dégradé à créer. Les valeurs acceptables sont `GradientType.LINEAR` ou `GradientType.RADIAL`.
- Le deuxième paramètre indique le tableau de valeurs colorimétriques à utiliser. Dans un dégradé linéaire, les couleurs sont organisées de gauche à droite. Dans un dégradé radial, les couleurs sont organisées de l'intérieur à l'extérieur. L'ordre des couleurs dans le tableau représente l'ordre dans lequel elles sont tracées dans le dégradé.
- Le troisième paramètre indique les valeurs de transparence alpha pour les couleurs correspondantes du paramètre précédent.
- Le quatrième paramètre spécifie les rapports, c'est-à-dire l'importance de chaque couleur dans le dégradé. Les valeurs acceptables vont de 0 à 255. Ces valeurs ne représentent pas une hauteur ou une largeur, mais plutôt la position au sein du dégradé : 0 représente le début du dégradé, et 255 la fin. Cette plage de rapports doit augmenter séquentiellement et comporter le même nombre d'éléments que les tableaux des couleurs et des valeurs alpha spécifiés comme deuxième et troisième paramètres.

Le cinquième paramètre, la matrice de transformation, est facultatif mais fréquemment utilisé, car il représente un moyen à la fois puissant et aisé de contrôler l'aspect du dégradé. Ce paramètre accepte une occurrence de l'objet Matrix. Le moyen le plus simple de créer un objet Matrix pour un dégradé consiste à utiliser la méthode `createGradientBox()` de la classe Matrix.

Définition d'un objet Matrix à utiliser avec un dégradé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez les méthodes `beginGradientFill()` et `lineGradientStyle()` de la classe flash.display.Graphics pour définir les dégradés à utiliser dans des formes. Lorsque vous définissez un dégradé, vous fournissez une matrice comme l'un des paramètres de ces méthodes.

La façon la plus facile de définir la matrice est d'utiliser la méthode `createGradientBox()`, de la classe Matrix, qui définit un tableau utilisé pour définir le dégradé. Définissez l'échelle, la rotation et la position du dégradé à l'aide des paramètres transmis à la méthode `createGradientBox()`. La méthode `createGradientBox()` reçoit les paramètres suivants :

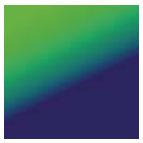
- Largeur de la zone de dégradé : largeur (en pixels) sur laquelle s'étend le dégradé
- Hauteur de la zone de dégradé : hauteur (en pixels) sur laquelle s'étend le dégradé
- Rotation de la zone de dégradé : rotation (en radians) qui sera appliquée au dégradé
- Translation horizontale : distance (en pixels) de déplacement horizontal du dégradé
- Translation verticale : distance (en pixels) de déplacement vertical du dégradé

Par exemple, supposons un dégradé possédant les caractéristiques suivantes :

- `GradientType.LINEAR`
- Deux couleurs, vert et bleu, avec le tableau `ratios` défini sur `[0, 255]`
- `SpreadMethod.PAD`
- `InterpolationMethod.LINEAR_RGB`

Les exemples suivants présentent des dégradés dans lesquels le paramètre `rotation` de la méthode `createGradientBox()` varie comme indiqué, mais tous les autres réglages restent inchangés :

`width = 100;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`
	

`width = 100;`
`height = 100;`
`rotation = Math.PI/4; // 45°`
`tx = 0;`
`ty = 0;`
	

`width = 100;`
`height = 100;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 0;`
	

Les exemples suivants présentent les effets sur un dégradé linéaire vert à bleu dans lequel les paramètres `rotation`, `tx` et `ty` de la méthode `createGradientBox()` varient comme indiqué, mais tous les autres réglages restent inchangés :

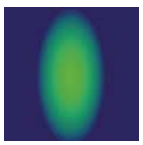
`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`
	

`width = 50;`
`height = 100;`
`rotation = 0`
`tx = 50;`
`ty = 0;`
	

`width = 100;`
`height = 50;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 0;`
	

`width = 100;`
`height = 50;`
`rotation = Math.PI/2; // 90°`
`tx = 0;`
`ty = 50;`
	

Les paramètres `width`, `height`, `tx` et `ty` de la méthode `createGradientBox()` ont une incidence sur la taille et la position d'un remplissage en dégradé *radial* également, comme indiqué dans l'exemple suivant :

`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 25;`
`ty = 0;`
	

Le code suivant produit le dernier dégradé radial illustré :

```
import flash.display.Shape;
import flash.display.GradientType;
import flash.geom.Matrix;

var type:String = GradientType.RADIAL;
var colors:Array = [0x00FF00, 0x000088];
var alphas:Array = [1, 1];
var ratios:Array = [0, 255];
var spreadMethod:String = SpreadMethod.PAD;
var interp:String = InterpolationMethod.LINEAR_RGB;
var focalPtRatio:Number = 0;

var matrix:Matrix = new Matrix();
var boxWidth:Number = 50;
var boxHeight:Number = 100;
var boxRotation:Number = Math.PI/2; // 90°
var tx:Number = 25;
var ty:Number = 0;
matrix.createGradientBox(boxWidth, boxHeight, boxRotation, tx, ty);

var square:Shape = new Shape;
square.graphics.beginGradientFill(type,
                           colors,
                           alphas,
                           ratios,
                           matrix,
                           spreadMethod,
                           interp,
                           focalPtRatio);
square.graphics.drawRect(0, 0, 100, 100);
addChild(square);
```

Notez que la largeur et la hauteur du dégradé sont déterminées par la largeur et la hauteur de la matrice du dégradé, plutôt que par celles qui sont dessinées à l'aide de l'objet Graphics. Si vous dessinez avec l'objet Graphics, vous tracez ce qui existe à ces coordonnées dans la matrice du dégradé. Même si vous utilisez l'une des méthodes de création de forme d'un objet de type Graphics, par exemple `drawRect()`, le dégradé n'est pas étiré en fonction de la taille de la forme dessinée : sa taille doit être spécifiée dans la matrice du dégradé.

L'exemple ci-dessous illustre la différence visuelle entre les dimensions de la matrice du dégradé et celles du dessin :

```
var myShape:Shape = new Shape();
var gradientBoxMatrix:Matrix = new Matrix();
gradientBoxMatrix.createGradientBox(100, 40, 0, 0, 0);
myShape.graphics.beginGradientFill(GradientType.LINEAR, [0xFF0000, 0x00FF00, 0x0000FF], [1,
1, 1], [0, 128, 255], gradientBoxMatrix);
myShape.graphics.drawRect(0, 0, 50, 40);
myShape.graphics.drawRect(0, 50, 100, 40);
myShape.graphics.drawRect(0, 100, 150, 40);
myShape.graphics.endFill();
this.addChild(myShape);
```

Ce code trace trois dégradés avec le même style de remplissage, spécifié avec une distribution égale de rouge, vert et bleu. Les dégradés sont tracés à l'aide de la méthode `drawRect()` avec des largeurs en pixels de 50, 100 et 150 respectivement. La matrice de dégradé qui est spécifiée dans la méthode `beginGradientFill()` est créée avec une largeur de 100 pixels. Le premier dégradé ne couvre donc que la moitié de son spectre, le deuxième le couvre en entier, et le troisième le couvre en entier et possède 50 pixels supplémentaires de bleu à droite.

La méthode `lineGradientStyle()` fonctionne de façon similaire à `beginGradientFill()`, si ce n'est qu'outre la définition du dégradé vous devez aussi indiquer l'épaisseur du trait à l'aide de la méthode `lineStyle()` avant de tracer. Le code suivant trace une boîte avec un trait dégradé rouge, vert et bleu :

```
var myShape:Shape = new Shape();
var gradientBoxMatrix:Matrix = new Matrix();
gradientBoxMatrix.createGradientBox(200, 40, 0, 0, 0);
myShape.graphics.lineStyle(5, 0);
myShape.graphics.lineGradientStyle(GradientType.LINEAR, [0xFF0000, 0x00FF00, 0x0000FF], [1,
1, 1], [0, 128, 255], gradientBoxMatrix);
myShape.graphics.drawRect(0, 0, 200, 40);
this.addChild(myShape);
```

Pour plus d'informations sur la classe Matrix, voir « Utilisation des objets Matrix » à la page 224.

Utilisation de la classe Math avec les méthodes de dessin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet Graphics trace des cercles et des carrés, mais il permet aussi de dessiner des formes plus complexes, en particulier lorsque les méthodes de dessin sont utilisées en combinaison avec les propriétés et les méthodes de la classe Math. La classe Math contient des constantes d'intérêt général en mathématiques, telles que `Math.PI` (environ 3,14159265...), qui est la constante définissant le rapport entre la circonférence et le diamètre d'un cercle. Elle contient également des méthodes de fonctions trigonométriques, entre autres `Math.sin()`, `Math.cos()` et `Math.tan()`. L'utilisation de ces méthodes et constantes pour le dessin de formes permet d'obtenir des effets visuels plus dynamiques, en particulier en utilisant des répétitions et des récursions.

De nombreuses méthodes de la classe Math attendent des mesures circulaires en radians, et non en degrés. La conversion entre ces deux types d'unités représente elle-même un cas d'utilisation courante de la classe Math :

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

L'exemple suivant crée une onde sinusoïdale et une onde cosinusoïdale, afin d'illustrer la différence entre les méthodes `Math.sin()` et `Math.cos()` pour une même valeur.

```
var sinWavePosition = 100;
var cosWavePosition = 200;
var sinWaveColor:uint = 0xFF0000;
var cosWaveColor:uint = 0x00FF00;
var waveMultiplier:Number = 10;
var waveStretcher:Number = 5;

var i:uint;
for(i = 1; i < stage.stageWidth; i++)
{
    var sinPosY:Number = Math.sin(i / waveStretcher) * waveMultiplier;
    var cosPosY:Number = Math.cos(i / waveStretcher) * waveMultiplier;

    graphics.beginFill(sinWaveColor);
    graphics.drawRect(i, sinWavePosition + sinPosY, 2, 2);
    graphics.beginFill(cosWaveColor);
    graphics.drawRect(i, cosWavePosition + cosPosY, 2, 2);
}
```

Animation avec l'API de dessin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'un des avantages de la création de contenu graphique avec l'API de dessin est que ce contenu peut être repositionné à loisir. Tout ce que vous tracez peut être modifié, en modifiant simplement les variables utilisées pour ce dessin. Vous pouvez donc obtenir de l'animation en changeant les variables et en retraçant, soit sur un nombre d'images donné, soit à l'aide d'un timer.

Par exemple, le code suivant change l'affichage à chaque nouvelle image (en écoutant l'événement `Event.ENTER_FRAME`) : il incrémente la valeur de degrés actuelle, puis ordonne à l'objet graphique d'effacer et redessiner avec la nouvelle position.

```
stage.frameRate = 31;

var currentDegrees:Number = 0;
var radius:Number = 40;
var satelliteRadius:Number = 6;

var container:Sprite = new Sprite();
container.x = stage.stageWidth / 2;
container.y = stage.stageHeight / 2;
addChild(container);
var satellite:Shape = new Shape();
container.addChild(satellite);

addEventListener(Event.ENTER_FRAME, doEveryFrame);

function doEveryFrame(event:Event):void
{
    currentDegrees += 4;
    var radians:Number = getRadians(currentDegrees);
    var posX:Number = Math.sin(radians) * radius;
    var posY:Number = Math.cos(radians) * radius;
    satellite.graphics.clear();
    satellite.graphics.beginFill(0);
    satellite.graphics.drawCircle(posX, posY, satelliteRadius);
}
function getRadians(degrees:Number):Number
{
return degrees * Math.PI / 180;
}
```

Pour produire un résultat nettement différent, vous pouvez expérimenter en modifiant les valeurs initiales au début du code, `currentDegrees`, `radius` et `satelliteRadius`. Par exemple, essayez en réduisant la valeur de la variable radius et/ou en augmentant la valeur de la variable totalSatellites. Ce n'est qu'un exemple simple de la façon dont l'API de dessin permet de créer du contenu visuel dont la complexité dissimule la simplicité de création.

Exemple d'utilisation de l'API de dessin : générateur algorithmique d'effets visuels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple de générateur algorithmique d'effets visuels trace de manière dynamique plusieurs « satellites », des cercles qui se déplacent suivant une orbite circulaire. Les fonctionnalités présentées sont les suivantes :

- Utilisation de l'API de dessin pour tracer une forme simple avec un aspect dynamique
- Utilisation de l'interaction de l'utilisateur pour modifier les propriétés utilisées pour le dessin
- Effet d'animation par effacement du contenu et retraçage à chaque nouvelle image.

L'exemple de la section précédente animait un satellite isolé à l'aide de l'événement `Event.ENTER_FRAME`. Cet exemple le reprend en y ajoutant un panneau de contrôle avec divers curseurs qui actualisent immédiatement l'affichage de plusieurs satellites. Dans cet exemple, le code est formalisé dans des classes externes et le code de création du satellite est imbriqué dans une boucle, en conservant une référence à chaque satellite dans le tableau `satellites`.

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application se trouvent dans le dossier Samples/AlgorithmicVisualGenerator. Celui-ci contient les fichiers suivants :

Fichier	Description
AlgorithmicVisualGenerator.fla	Fichier d'application principal de Flash Professional (FLA)
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	Classe comportant les principales fonctionnalités de l'application : dessin des satellites sur la scène et actualisation des variables utilisées pour ces dessins en réponse aux événements du panneau de contrôle.
com/example/programmingas3/algorithmic/ControlPanel.as	Cette classe gère les interactions de l'utilisateur avec les curseurs et distribue les événements appropriés.
com/example/programmingas3/algorithmic/Satellite.as	Cette classe représente l'objet d'affichage qui tourne sur son orbite autour d'un point central et contient des propriétés relatives à son état de dessin actuel.

Définition des écouteurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application commence par créer trois écouteurs. Le premier attend qu'un événement soit distribué par le panneau de contrôle pour signaler qu'une reconstruction des satellites est nécessaire. Le second attend des changements de taille de la scène du fichier SWF. Le troisième attend le passage de chaque image du fichier SWF et son retraçage à l'aide de la fonction `doEveryFrame()`.

Création des satellites

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Après la définition de ces écouteurs, la fonction `build()` est appelée. Cette fonction appelle d'abord la fonction `clear()`, qui efface le contenu du tableau `satellites` et supprime les éventuelles formes déjà présentes sur la scène. Cette précaution est nécessaire car la fonction `build()` peut être appelée à nouveau à la suite d'un événement diffusé par le panneau de contrôle, par exemple en cas de modification des couleurs. Dans ce cas, les satellites doivent être supprimés et recréés.

La fonction crée ensuite les satellites, en définissant les propriétés initiales nécessaires à cette création, dont la variable `position` qui définit une position aléatoire sur l'orbite et la variable `color`, qui dans cet exemple ne change pas une fois que le satellite a été créé.

Lors de la création de chaque satellite, une référence est ajoutée dans le tableau `satellites`. Lorsque la fonction `doEveryFrame()` est appelée, elle actualise tous les satellites du tableau.

Actualisation de la position des satellites

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La fonction `doEveryFrame()` est au cœur du processus d'animation de l'application. Elle est appelée à chaque image, donc à une fréquence identique à la cadence du fichier SWF. Les variables du dessin changent légèrement, ce qui permet d'obtenir un effet d'animation.

La fonction efface d'abord tous les éventuels dessins antérieurs et retrace l'arrière-plan. Elle effectue ensuite une boucle dans le conteneur de chaque satellite et incrémente la propriété `position` de chaque satellite, puis actualise les propriétés `radius` et `orbitRadius` qui peuvent avoir été modifiées par suite d'une action de l'utilisateur dans le panneau de contrôle. Enfin, la nouvelle position de chaque satellite est affichée en appelant la méthode `draw()` de la classe Satellite.

Notez que la variable de compteur `i` n'est incrémentée que jusqu'à la valeur de la variable `visibleSatellites`. En effet, si l'utilisateur a limité à l'aide du panneau de contrôle le nombre de satellites affichés, les autres satellites de la boucle ne doivent pas être redessinés, mais masqués. La boucle chargée de cette action suit immédiatement celle qui est responsable du dessin.

Lorsque la fonction doEveryFrame() se termine, le nombre de satellites visibles (`visibleSatellites`) est redessiné aux nouvelles positions.

Réponse aux interactions de l'utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'interactivité est assurée par le panneau de contrôle, qui est géré par la classe ControlPanel. Cette classe définit pour chaque curseur un écouteur et des valeurs individuelles minimum, maximum et par défaut. Lorsque l'utilisateur déplace ces curseurs, la fonction `changeSetting()` est appelée. Elle actualise les propriétés du panneau de contrôle. Si la modification nécessite un nouvel affichage, un événement est distribué et pris en charge dans le fichier principal de l'application. En fonction de la modification signalée par le panneau de contrôle, la fonction `doEveryFrame()` redessine chaque satellite sur la base des nouvelles variables.

Améliorations possibles

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple est simplement destiné à illustrer la création d'effets visuels avec l'API de dessin. Il utilise relativement peu de lignes de code pour créer une animation interactive qui semble très complexe. Même ainsi, cet exemple pourrait être amélioré moyennant quelques modifications mineures. Voici quelques idées :

- La fonction `doEveryFrame()` pourrait incrémenter la valeur colorimétrique du satellite.
- La fonction `doEveryFrame()` pourrait réduire ou augmenter progressivement le rayon de l'orbite du satellite.
- Il n'est pas nécessaire que l'orbite du satellite soit circulaire, la classe Math permet de le déplacer selon une sinusoïdale, par exemple.
- Les satellites pourraient utiliser une détection de collision entre eux.

L'API de dessin peut être considérée comme autre solution pour la création d'effets visuels dans l'environnement de création Flash, en dessinant des formes de base lors de l'exécution. Mais elle permet aussi de créer des effets visuels qu'il serait impossible de créer manuellement. L'API de dessin et quelques notions de mathématiques permettent au développeur en ActionScript de donner vie à de nombreuses créations inattendues.

Utilisation avancée de l'API de dessin

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Flash Player 10, Adobe AIR 1.5 et les moteurs d'exécution Flash ultérieurs prennent en charge un jeu avancé de fonctions de dessin. Les améliorations de l'API de dessin associées à ces moteurs d'exécution, qui étendent les méthodes de dessin des versions antérieures, vous permettent de définir des ensembles de données pour générer des formes, les modifier lors de l'exécution et créer des effets tridimensionnels. Les améliorations de l'API de dessin consolident les méthodes existantes comme commandes alternatives. Ces commandes s'appuient sur des tableaux de vecteurs et des classes d'énumération pour fournir des ensembles de données aux méthodes de dessin. Les tableaux de vecteurs accélèrent le rendu de formes plus complexes. Les développeurs peuvent modifier les valeurs des tableaux par programmation pour rendre des formes dynamiques à l'exécution.

Les nouvelles fonctions de dessins de Flash Player 10 sont décrites dans les sections suivantes : « Tracés de dessin » à la page 244, « Définition des règles d'enroulement » à la page 245, « Utilisation des classes de données graphiques » à la page 247 et « A propos de l'utilisation de drawTriangles() » à la page 250.

Vous souhaiterez probablement effectuer les tâches suivantes à l'aide des fonctions avancées de l'API de dessin dans ActionScript :

- Stockage des données destinées aux méthodes de dessin à l'aide d'objets Vector
- Définition de tracés pour tracer des formes par programmation en une seule opération
- Définition de règles d'enroulement pour déterminer le remplissage de formes se chevauchant
- La lecture du contenu graphique vectoriel d'un objet d'affichage, par exemple pour sérialiser et enregistrer les données graphiques, pour générer une feuille Sprite à l'exécution ou pour dessiner une copie du contenu graphique vectoriel.
- Utilisation de triangles et de méthodes de dessin pour obtenir des effets tridimensionnels

Concepts importants et terminologie

La liste de référence suivante énumère les termes importants que vous rencontrerez dans cette section :

- Vecteur : tableau de valeurs d'un type de données identique. Un objet Vector peut stocker un tableau de valeurs qu'utilisent des méthodes de dessin pour construire des lignes et des formes par le biais d'une commande unique. Pour plus d'informations sur les objets Vector, voir « Tableaux indexés » à la page 27.
- Tracé : un tracé est composé d'un ou de plusieurs segments droits ou incurvés. Le début et la fin de chaque segment sont indiqués par des coordonnées qui fonctionnent à la manière d'épingles maintenant un fil en place. Un tracé peut être fermé (un cercle, par exemple) ou ouvert, s'il comporte des extrémités distinctes (une ligne onduleuse, par exemple).
- Enroulement : direction d'un tracé tel qu'il est interprété par le rendu, soit positive (sens horaire) soit négative (sens antihoraire).
- GraphicsStroke : classe permettant de définir le style de ligne. Bien que le « trait » à proprement parler n'ait pas été inclus dans la nouvelle API de dessin, l'utilisation d'une classe pour désigner un style de ligne avec ses propres propriétés de remplissage constitue l'une des améliorations intégrées. Vous pouvez régler dynamiquement le style d'une ligne à l'aide de la classe GraphicsStroke.
- Objet Fill : objet créé à l'aide de classes d'affichage telles que flash.display.GraphicsBitmapFill et flash.display.GraphicsGradientFill, et transmis à la commande de dessin `Graphics.drawGraphicsData()`. Les objets Fill et les commandes de dessin optimisées proposent une approche de programmation plus orientée objets pour répliquer `Graphics.beginBitmapFill()` et `Graphics.beginGradientFill()`.

Tracés de dessin

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La section sur le dessin de lignes et de courbes (voir « Dessin de lignes et de courbes » à la page 231) a présenté les commandes permettant de tracer une ligne (`Graphics.lineTo()`) ou courbe (`Graphics.curveTo()`) unique, puis de la déplacer vers un autre point (`Graphics.moveTo()`) pour obtenir une forme. Les méthodes `Graphics.drawPath()` et `Graphics.drawTriangles()` acceptent une série d'objets représentant ces mêmes commandes de dessin comme paramètre. Ces méthodes permettent de définir une série de commandes `Graphics.lineTo()`, `Graphics.curveTo()` ou `Graphics.moveTo()` traitées par le moteur d'exécution Flash en une seule instruction.

La classe d'énumération GraphicsPathCommand définit une série de constantes qui correspondent aux commandes de dessin. Vous transmettez une série de ces constantes (encapsulées dans une occurrence de Vector) comme paramètre de la méthode `Graphics.drawPath()`. Vous pouvez ensuite rendre une forme entière ou plusieurs formes à l'aide d'une seule commande. Vous pouvez aussi modifier les valeurs transmises à ces méthodes pour modifier une forme existante.

Outre l'occurrence Vector des commandes de dessin, la méthode `drawPath()` a besoin d'un ensemble de coordonnées qui correspondent aux coordonnées de chaque commande de dessin. Créez une occurrence de Vector contenant des coordonnées (instances de Number) et transmettez-la à la méthode `drawPath()` comme second argument (`data`).

Remarque : *les valeurs du vecteur ne sont pas des objets Point. Le vecteur est une série de nombres, dont chaque paire représente une paire de coordonnées x/y.*

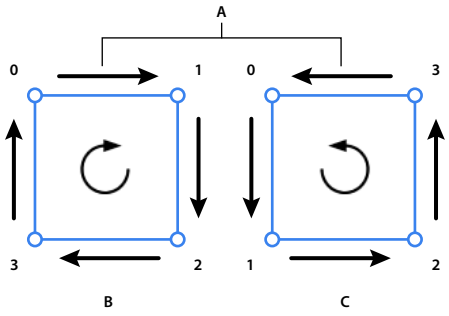
La méthode `Graphics.drawPath()` fait correspondre chaque commande à ses valeurs de point respectives (une série de deux ou quatre nombres) pour générer un tracé dans l'objet Graphics :

```
package
{
    import flash.display.*;

    public class DrawPathExample extends Sprite
    {
        public function DrawPathExample(){

            var squareCommands:Vector.<int> = new Vector.<int>(5, true);
            squareCommands[0] = GraphicsPathCommand.MOVE_TO;
            squareCommands[1] = GraphicsPathCommand.LINE_TO;
            squareCommands[2] = GraphicsPathCommand.LINE_TO;
            squareCommands[3] = GraphicsPathCommand.LINE_TO;
            squareCommands[4] = GraphicsPathCommand.LINE_TO;

            var squareCoord:Vector.<Number> = new Vector.<Number>(10, true);
            squareCoord[0] = 20; //x
            squareCoord[1] = 10; //y
            squareCoord[2] = 50;
            squareCoord[3] = 10;
            squareCoord[4] = 50;
            squareCoord[5] = 40;
            squareCoord[6] = 20;
            squareCoord[7] = 40;
            squareCoord[8] = 20;
            squareCoord[9] = 10;

            graphics.beginFill(0x442266);//set the color
            graphics.drawPath(squareCommands, squareCoord);
        }
    }
}
```

Définition des règles d'enroulement

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'API de dessin améliorée prend désormais en charge le concept d'« enroulement » de tracé, qui définit la direction de ce dernier. L'enroulement d'un tracé est soit positif (sens horaire) soit négatif (sens antihoraire). L'ordre dans lequel le rendu interprète les coordonnées que fournit le vecteur au paramètre data détermine l'enroulement.



Enroulement positif et négatif
***A.** Les flèches indiquent la direction du tracé.* ***B.** Enroulement positif (sens horaire)* ***C.** Enroulement négatif (sens antihoraire)*

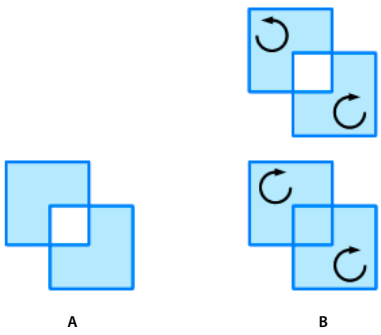
En outre, vous remarquerez que la méthode `Graphics.drawPath()` possède un troisième paramètre facultatif, appelé « winding » :

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

Dans ce contexte, le troisième paramètre est une chaîne ou une constante qui détermine la règle d'enroulement ou de remplissage de tracés se croisant (les valeurs de constante sont définies dans la classe GraphicsPathWinding sous la forme `GraphicsPathWinding.EVEN_ODD` ou `GraphicsPathWinding.NON_ZERO`). La règle d'enroulement est importante lorsque des tracés se croisent.

Règle d'enroulement standard, la règle pair-impair est utilisée par l'API de dessin héritée. C'est aussi la règle par défaut de la méthode `Graphics.drawPath()`. Lorsqu'elle est appliquée, des tracés qui se croisent alternent entre des remplissages ouverts et fermés. Si deux carrés utilisant un même remplissage se croisent, la zone d'intersection est remplie. En règle générale, les zones adjacentes ne sont pas remplies ou leur remplissage n'est pas effacé.

La règle non null, en revanche, se fonde sur l'enroulement (direction du tracé) pour déterminer si les zones définies par des tracés qui se croisent sont remplies. Lorsque des tracés dont l'enroulement est différent se croisent, la zone définie n'est pas remplie, comme avec la règle pair-impair. Si l'enroulement est identique, la zone dont le remplissage serait effacé est remplie :

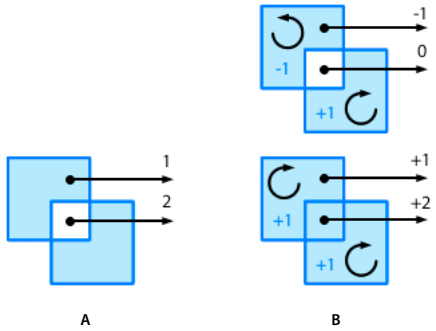


Règles d'enroulement associée aux zones d'intersection
***A.** Règle d'enroulement pair-impair **B.** Règle d'enroulement non nul*

Nom des règles d'enroulement

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les noms se réfèrent à une règle beaucoup plus spécifique qui définit comment ces remplissages sont gérés. On affecte aux chemins d'enroulement positifs une valeur +1 et aux négatifs une valeur -1. A partir d'un point au sein d'une surface close d'une forme, tirez un trait qui s'étend indéfiniment. Le nombre de fois que cette ligne traverse un chemin, ainsi que les valeurs combinées de ces chemins, est utilisé pour déterminer le remplissage. Pour des enroulements pair-impair, c'est le nombre de fois que la ligne traverse un chemin qui est utilisé. Lorsque le décompte est un nombre impair, la zone est remplie. Lorsqu'il est impair, la zone ne l'est pas. Pour des enroulements non nuls, ce sont les valeurs affectées aux chemins qui sont utilisées. Lorsque les valeurs combinées du chemin ne sont pas nulles, la zone est remplie. Lorsque la valeur combinée est 0, la zone n'est pas remplie.



Règles de décompte et remplissage d'enroulements
***A.** Règle d'enroulement pair-impair **B.** Règle d'enroulement non nul*

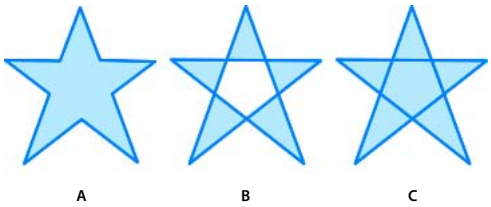
Utilisation des règles d'enroulement

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Ces règles de remplissage sont complexes mais elles s'avèrent nécessaires dans certaines situations. Par exemple, prenons le dessin d'une forme étoilée. Suivant la règle pair-impair standard, la forme nécessiterait dix lignes différentes. Suivant la règle d'enroulement non nul, ces dix lignes sont réduites à cinq. Voici le code ActionScript pour une étoile à cinq lignes et une règle d'enroulement non nul :

```
graphics.beginFill(0x60A0FF);
graphics.drawPath( Vector.<int>([1,2,2,2,2]),  Vector.<Number>([66,10, 23,127, 122,50, 10,49,
109,127]),   GraphicsPathWinding.NON_ZERO);
```

Et voici la forme de l'étoile :



Une forme d'étoile qui utilise différentes règles d'enroulement
***A.** 10 lignes en pair-impair **B.** 5 lignes en pair-impair **C.** 5 lignes non nulles*

Et, comme les images sont animées ou utilisées comme textures sur des objets à trois dimensions et qu'elles se recouvrent, les règles d'enroulement deviennent plus importantes.

Utilisation des classes de données graphiques

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'API de dessin amélioré inclut un ensemble de classes dans le package flash.display qui met en œuvre l'interface IGraphicsData. Ces classes fonctionnent comme des objets de valeur (conteneurs de données) qui représentent les méthodes de l'API de dessin.

Les classes suivantes implémentent l'interface IGraphicsData :

- GraphicsBitmapFill
- GraphicsEndFill

- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

Ces classes vous permettent de stocker un dessin complet dans un objet Vector de type IGraphicsData (Vector.<IGraphicsData>). Vous pouvez ensuite réutiliser les données graphiques comme source de données pour les autres instances de la forme ou pour stocker les informations de dessin pour une utilisation ultérieure.

Vous remarquerez qu'il existe plusieurs classes de remplissage (Fill) correspondant à chaque style de remplissage, mais une seule classe de trait (Stroke). ActionScript propose une seule classe de trait IGraphicsData car elle définit son style sur la base des classes de remplissage. Ainsi, chaque trait est en fait défini par une combinaison de la classe stroke et d'une classe fill. Pour le reste, l'API de ces classes de données graphiques est identique aux méthodes qu'elles représentent dans la classe flash.display.Graphics :

Méthode graphique	**Classe correspondante**
beginBitmapFill()	GraphicsBitmapFill
beginFill()	GraphicsSolidFill
beginGradientFill()	GraphicsGradientFill
beginShaderFill()	GraphicsShaderFill
lineBitmapStyle()	GraphicsStroke + GraphicsBitmapFill
lineGradientStyle()	GraphicsStroke + GraphicsGradientFill
lineShaderStyle()	GraphicsStroke + GraphicsShaderFill
lineStyle()	GraphicsStroke + GraphicsSolidFill
moveTo() lineTo() curveTo() drawPath()	GraphicsPath
drawTriangles()	GraphicsTrianglePath

En outre, la classe GraphicsPath possède ses propres méthodes d'utilitaire (`GraphicsPath.moveTo()`, `GraphicsPath.lineTo()`, `GraphicsPath.curveTo()`, `GraphicsPath.wideLineTo()` et `GraphicsPath.wideMoveTo()`) pour simplifier la définition de ces commandes pour une occurrence de GraphicsPath. Ces méthodes facilitent la définition ou la mise à jour directe des commandes et des valeurs de données.

Réalisation d'un dessin avec les données graphiques vectorielles

Lorsque vous disposez d'une série d'instances de IGraphicsData, utilisez la méthode `drawGraphicsData()` de la classe Graphics pour effectuer le rendu des graphiques. La méthode `drawGraphicsData()` exécute un ensemble d'instructions de dessin à partir d'un vecteur d'instances de IGraphicsData en séquence :

```
// stroke object
var stroke:GraphicsStroke = new GraphicsStroke(3);
stroke.joints = JointStyle.MITER;
stroke.fill = new GraphicsSolidFill(0x102020);// solid stroke

// fill object
var fill:GraphicsGradientFill = new GraphicsGradientFill();
fill.colors = [0x0000FF, 0xEEFFEE];
fill.matrix = new Matrix();
fill.matrix.createGradientBox(70, 70, Math.PI/2);
// path object
var path:GraphicsPath = new GraphicsPath(new Vector.<int>(), new Vector.<Number>());
path.commands.push(GraphicsPathCommand.MOVE_TO, GraphicsPathCommand.LINE_TO,
GraphicsPathCommand.LINE_TO);
path.data.push(125,0, 50,100, 175,0);

// combine objects for complete drawing
var drawing:Vector.<IGraphicsData> = new Vector.<IGraphicsData>();
drawing.push(stroke, fill, path);

// draw the drawing
graphics.drawGraphicsData(drawing);
```

En modifiant une valeur du tracé utilisé par le dessin de l'exemple, il est possible de retracer la forme plusieurs fois pour obtenir une image plus complexe :

```
// draw the drawing multiple times
// change one value to modify each variation
graphics.drawGraphicsData(drawing);
path.data[2] += 200;
graphics.drawGraphicsData(drawing);
path.data[2] -= 150;
graphics.drawGraphicsData(drawing);
path.data[2] += 100;
graphics.drawGraphicsData(drawing);
path.data[2] -= 50;graphicsS.drawGraphicsData(drawing);
```

Bien que les objets IGraphicsData puissent définir des styles de remplissage et de trait, ceux-ci ne sont pas obligatoires. Autrement dit, il est possible de définir des styles à l'aide des méthodes de la classe Graphics ou d'utiliser des objets IGraphicsData pour tracer un ensemble enregistré de tracés, et inversement.

Remarque : *la méthode* `Graphics.clear()` *permet d'effacer un dessin avant d'en commencer un nouveau, à moins que vous ne complétiez le dessin d'origine, comme l'illustre l'exemple ci-dessus. Lorsque vous modifiez une partie d'un tracé ou d'un ensemble d'objets IGraphicsData, retracez le dessin entier pour visualiser les changements.*

Lorsque vous utilisez des classes de données graphiques, le remplissage est rendu chaque fois que trois points au moins sont tracés car la forme est nécessairement fermée à ce point. Bien que le remplissage ait un effet de fermeture, ce n'est pas le cas du trait. Ce comportement est différent de celui de commandes `Graphics.lineTo()` ou `Graphics.moveTo()` utilisées plusieurs fois.

Lecture de données graphiques vectorielles

Flash Player 11.6 et les versions ultérieures, Adobe AIR 3.6 et les versions ultérieures

Dans Flash Player 11.6, Adobe AIR 3.6 et versions ultérieures, vous pouvez utiliser la méthode `readGraphicsData()` de la classe Graphics non seulement pour tracer le contenu vectoriel d'un objet d'affichage, mais aussi pour obtenir une représentation en données du contenu graphique vectoriel de cet objet. Ceci peut être utilisé pour créer un instantané d'un graphique afin notamment d'enregistrer, de copier ou de créer une feuille Sprite lors de l'exécution.

L'appel de la méthode `readGraphicsData()` renvoie une occurrence de Vector contenant des objets IGraphicsData. Il s'agit des mêmes objets que ceux utilisés pour tracer des graphiques vectoriels à l'aide de la méthode `drawGraphicsData()`.

Plusieurs limitations s'appliquent à la lecture des graphiques vectoriels avec la méthode `readGraphicsData()`. Pour plus d'informations, voir l'entrée readGraphicsData() dans le Guide de référence ActionScript.

A propos de l'utilisation de drawTriangles()

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Une autre méthode avancée intégrée à Flash Player 10 et Adobe AIR 1.5, `Graphics.drawTriangles()`, s'apparente à la méthode `Graphics.drawPath()`. La méthode `Graphics.drawTriangles()` utilise également un objet Vector.<Number> pour spécifier les points permettant de dessiner un tracé.

La méthode `Graphics.drawTriangles()` a néanmoins pour but principal de faciliter la création d'effets tridimensionnels par le biais d'ActionScript. Pour plus d'informations sur l'utilisation de `Graphics.drawTriangles()` pour produire des effets tridimensionnels, voir « Création d'effets 3D à l'aide de triangles » à la page 375.

Chapitre 13 : Utilisation des images bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Outre ses possibilités en matière de dessin vectoriel, ActionScript 3.0 permet également de créer des images bitmap ou de manipuler les données de pixels d'images bitmap externes chargées dans un fichier SWF. Cette possibilité de lire et modifier des valeurs de pixel individuelles permet de créer des effets visuels comme ceux des filtres et d'utiliser les fonctions intégrées de gestion du « bruit » pour créer des textures et des bruits aléatoires.

- Renaun Erickson: Rendering game assets in ActionScript using blitting techniques (disponible en anglais uniquement)
- Bitmap programming: chapitre 26 du document Essential ActionScript 3, par Colin Moock (O'Reilly Media, 2007, disponible en anglais uniquement)
- Mike Jones: Working with Sprites in Pushbutton Engine (disponible en anglais uniquement)
- Flash & Math: Pixel Particles Made Simple (disponible en anglais uniquement)
- Flixel

Principes de base de l'utilisation des images bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tout travail avec des images numériques nécessite de gérer deux types de graphismes : les bitmaps et les graphismes vectoriels. Les images bitmap, également appelées graphismes en points, sont composées de petits carrés (les pixels) organisés en une grille rectangulaire. De leur côté, les graphismes vectoriels sont composés de formes géométriques (lignes, courbes et polygones) générées à l'aide de formules mathématiques.

Les images bitmap sont définies par la largeur et la hauteur de l'image, mesurées en pixels, et par le nombre de bits contenu dans chaque pixel (ce nombre de bits définit le nombre de couleurs que peut comporter l'image). Dans le cas d'une image bitmap utilisant le modèle colorimétrique RVB, les pixels sont composés de trois octets : rouge, vert et bleu. Chaque octet contient une valeur comprise entre 0 et 255. C'est la combinaison de ces octets pour chaque pixel qui produit une couleur, un peu comme le mélange des couleurs de base par un peintre. Par exemple, un pixel contenant les valeurs 255, 102 et 0 respectivement pour les octets dévolus au rouge, au vert et au bleu produira un orange vif.

La qualité d'une image bitmap est déterminée en combinant la résolution en pixels de l'image avec sa profondeur de couleur exprimée en bits ou en octets. La *résolution* définit le nombre de pixels contenus dans l'image. Plus le nombre de pixels est important, plus la résolution est élevée et plus l'image semble bien définie. La *profondeur de couleurs* définit la quantité d'informations colorimétriques contenues par chaque pixel. Par exemple, une image ayant une profondeur de couleur de 16 bits par pixel ne peut pas représenter un nombre de nuances aussi élevé qu'une image ayant une profondeur de couleur de 48 bits. En conséquence, l'image sur 48 bits aura plus de nuances que la version sur 16 bits.

Dans la mesure où les images bitmap dépendent de la résolution, il est délicat de modifier leur échelle, ce qui se remarque particulièrement avec les images bitmap agrandies, qui perdent beaucoup de détails et de qualité.

Format des fichiers bitmap

Les images bitmap existent en divers formats de fichier. Ces formats utilisent différents types d'algorithmes de compression pour réduire la taille des fichiers et optimiser la qualité de l'image en fonction de sa destination. Les moteurs d'exécution Adobe gèrent les formats d'image bitmap BMP, GIF, JPG, PNG et TIFF.

BMP

Le format BMP (pixellisé) est un format d'image par défaut utilisé par le système d'exploitation Microsoft Windows. Il ne fait appel à aucune forme d'algorithme de compression ; il en résulte généralement des tailles de fichiers assez importantes.

GIF

Le format GIF (Graphics Interchange Format) fut développé à l'origine par CompuServe en 1987, dans le but de transmettre des images en 256 couleurs (codées sur 8 bits). Ce format, qui permet d'obtenir des fichiers de petite taille, est très utilisé pour les images sur le Web. Toutefois, en raison de son nombre limité de couleurs, ce format n'est pas adapté aux photographies, qui nécessitent en général un nombre de nuances plus élevé. Les images GIF autorisent la transparence sur un bit, ce qui permet de rendre les couleurs invisibles (ou transparentes). C'est ainsi que sont produits les fonds transparents des images de pages Web.

JPEG

Développé par le Joint Photographic Experts Group (JPEG), le format d'image JPEG (souvent noté JPG) fait appel à un algorithme de compression avec perte pour autoriser une profondeur de couleur de 24 bits avec une taille de fichier très réduite. En raison de la compression avec perte, chaque fois que l'image est enregistrée elle perd des données, donc de la qualité, mais la taille de fichier résultante en est d'autant plus réduite. Le format JPEG est idéal pour les photographies, car il permet d'afficher des millions de couleurs différentes. La possibilité de contrôler le taux de compression appliqué à l'image permet de modifier la qualité de l'image et la taille du fichier.

PNG

Le format PNG (Portable Network Graphics) a été créé comme autre possibilité gratuite (open source) au format de fichier GIF, qui est breveté. Les fichiers PNG acceptent jusqu'à 64 bits de profondeur de couleur, ce qui permet d'obtenir jusqu'à 16 millions de couleurs. Le format PNG étant relativement récent, les versions anciennes de certains navigateurs ne gèrent pas ces fichiers. Contrairement au JPG, le format PNG utilise une compression sans perte, ce qui signifie qu'aucune donnée de l'image n'est perdue lors de l'enregistrement. De plus, les fichiers PNG acceptent également la transparence alpha, ce qui autorise jusqu'à 256 niveaux de transparence.

TIFF

Le format TIFF (ou Tagged Image File Format) était le format multiplate-forme de choix avant l'arrivée de PNG. Le format TIFF a un inconvénient qui est la multiplicité de ses variétés : aucun lecteur n'est en mesure de traiter toutes les versions disponibles. De surcroît, les navigateurs Web ne prennent pas en charge ce format actuellement. TIFF peut utiliser une compression avec ou sans pertes et il est en mesure de traiter des espaces de couleurs spécifiques au périphérique tels que CMJN (cyan-magenta-jaune-noir).

Fichiers bitmap transparents et opaques

Les images bitmap qui utilisent le format GIF ou PNG peuvent comporter pour chaque pixel un octet supplémentaire pour le canal alpha. Cet octet de pixel supplémentaire représente la valeur de transparence du pixel.

Les images GIF n'autorisent la transparence que sur la base d'un bit, ce qui ne permet de spécifier la transparence que pour une seule couleur (dans une palette de 256 couleurs). Par contre, les images PNG acceptent jusqu'à 256 niveaux de transparence. Cette caractéristique est particulièrement intéressante pour « fondre » les images ou le texte avec l'arrière-plan.

ActionScript 3.0 permet de gérer cet octet supplémentaire de transparence à l'aide de la classe BitmapData. Comme le modèle de transparence PNG, ActionScript propose jusqu'à 256 niveaux de transparence.

Concepts importants et terminologie

La liste suivante contient des termes importants relatifs aux graphiques bitmap :

Alpha Niveau de transparence (ou, plus précisément, d'opacité) d'une couleur ou d'une image. La valeur alpha est fréquemment appelée valeur du *canal alpha*.

Couleur ARGB Modèle colorimétrique suivant lequel la couleur de chaque pixel est définie selon ses composants rouge, vert et bleu, ainsi que par sa transparence spécifiée comme valeur alpha.

Canal de couleur En général, les couleurs sont représentées par le mélange des couleurs primaires, rouge, vert et bleu (pour les graphiques sur ordinateur). Chaque couleur primaire est considérée comme un canal de couleur. C'est le mélange des proportions de chaque canal de couleur qui détermine la couleur finale.

Codage des couleurs Parfois appelée *codage*, cette caractéristique détermine la quantité de mémoire vive dévolue à chaque pixel, ce qui, indirectement, définit le nombre de couleurs qu'il est possible d'afficher dans l'image.

Pixel Unité d'information de base d'une image bitmap (fondamentalement, un point coloré).

Résolution Dimensions d'une image en pixels, qui détermine le niveau de détails fins d'une image. La résolution est fréquemment exprimée en nombre de pixels pour la largeur et la hauteur.

Couleur RVB Modèle colorimétrique suivant lequel la couleur de chaque pixel est définie selon ses composants rouge, vert et bleu.

Classes Bitmap et BitmapData

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les principales classes ActionScript 3.0 d'utilisation des images bitmap sont la classe Bitmap, qui permet d'afficher les images bitmap à l'écran, et la classe BitmapData, qui permet d'accéder et de manipuler les données brutes d'une image bitmap.

Voir aussi

flash.display.Bitmap

flash.display.BitmapData

Présentation de la classe Bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Sous-classe de la classe DisplayObject, la classe Bitmap est la principale classe utilisée en ActionScript 3.0 pour l'affichage d'images bitmap. Ces images sont souvent chargées par le biais de la classe flash.display.Loader ou créées dynamiquement à l'aide du constructeur `Bitmap()`. En cas de chargement d'une image provenant d'une source externe, un objet Bitmap ne peut contenir que des images aux formats GIF, JPEG ou PNG. L'occurrence de l'objet Bitmap peut être considérée comme enveloppe d'un objet BitmapData devant être affiché sur la scène. Une occurrence de Bitmap étant un objet d'affichage, toutes les caractéristiques et fonctionnalités des objets d'affichage peuvent être utilisées pour la manipuler. Pour plus d'informations sur l'utilisation des objets d'affichage, voir « Programmation de l'affichage » à la page 156.

Accrochage et lissage des pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Outre les fonctionnalités communes à tous les objets d'affichage, la classe Bitmap dispose de quelques fonctionnalités supplémentaires, propres aux images bitmap.

La propriété `pixelSnapping` de la classe Bitmap détermine si un objet Bitmap accroche le pixel le plus proche. Cette propriété accepte l'une des trois constantes définies dans la classe PixelSnapping : `ALWAYS`, `AUTO` et `NEVER`.

La syntaxe de l'accrochage aux pixels est la suivante :

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

Lorsque des images bitmap sont redimensionnées, il est fréquent qu'elles deviennent floues et distordues. Pour réduire cette distorsion, utilisez la propriété `smoothing` de la classe BitmapData. Lorsque cette propriété booléenne a la valeur `true`, elle adoucit les pixels par un effet d'anticrènelage appliqué aux images redimensionnées. Celles-ci ont alors un aspect plus clair, plus naturel.

Présentation de la classe BitmapData

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe BitmapData, qui se trouve dans le package flash.display, peut être comparée à un cliché photographique des pixels d'une image bitmap, que celle-ci soit chargée ou créée dynamiquement. Ce cliché est représenté par une grille de données des pixels de l'objet. La classe BitmapData contient également une série de méthodes permettant de créer et modifier des données de pixels.

Pour instancier un objet BitmapData, utilisez le code suivant :

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

Les paramètres `width` et `height` déterminent la taille de l'image bitmap. Dans AIR 3 et Flash Player 11 et les versions ultérieures, les limites de taille pour un objet BitmapData ont été supprimées. La taille maximale d'une image bitmap dépend du système d'exploitation.

Dans AIR 1.5 et Flash Player 10, la taille maximale d'un objet BitmapData est de 8 191 pixels en largeur ou en hauteur, et le nombre total de pixels ne peut pas excéder 16 777 215 pixels (ainsi, si la largeur d'un objet BitmapData est de 8 191 pixels, sa hauteur maximale doit être de 2 048 pixels). Dans Flash Player 9 et les versions antérieures, ainsi que dans AIR 1.1 et les versions antérieures, la limite est de 2 880 pixels de haut sur 2 880 pixels de large.

Le paramètre `transparent` indique si l'image bitmap comporte un canal alpha (`true`) ou non (`false`). Le paramètre `fillColor` est une valeur colorimétrique sur 32 bits qui spécifie la couleur d'arrière-plan, ainsi que la valeur de la transparence (si cette dernière est `true`). L'exemple suivant crée un objet BitmapData dont l'arrière-plan orange est transparent à 50 % :

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

Pour afficher un objet BitmapData nouvellement créé, affectez-le à une occurrence de Bitmap. Pour ce faire, vous pouvez soit transmettre l'objet BitmapData sous forme de paramètre du constructeur de l'objet Bitmap, ou l'affecter à la propriété `bitmapData` d'une occurrence de Bitmap existante. Vous devez également affecter cette occurrence de Bitmap à la liste d'affichage en appelant la méthode `addChild()` ou `addChildAt()` du conteneur d'objet d'affichage qui contiendra cette occurrence de Bitmap. Pour plus d'informations sur l'utilisation de la liste d'affichage, voir « Ajout d'objets d'affichage à la liste d'affichage » à la page 165.

L'exemple suivant crée un objet BitmapData avec un remplissage rouge, et l'affiche dans une occurrence de Bitmap :

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

Manipulation des pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe BitmapData contient des méthodes qui permettent de modifier les valeurs des données de pixels.

Manipulation individuelle de pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour modifier une image bitmap au niveau des pixels, il est d'abord nécessaire d'obtenir les valeurs colorimétriques des pixels de la zone à modifier. La méthode `getPixel()` permet d'obtenir ces valeurs de pixels.

La méthode `getPixel()` renvoie les valeurs RVB des coordonnées (de pixels) x et y qui lui sont passées en paramètres. Si l'un des pixels comporte des informations de transparence (canal alpha), il est nécessaire d'utiliser la méthode `getPixel32()`. Cette méthode récupère également une valeur RVB, mais contrairement à ce qui se passe avec `getPixel()`, la valeur renvoyée par `getPixel32()` contient des données supplémentaires qui représentent la valeur du canal alpha (transparence) du pixel sélectionné.

Pour simplement modifier la couleur ou la transparence d'un pixel contenu dans un bitmap, il est aussi possible d'utiliser la méthode `setPixel()` ou `setPixel32()`. Pour définir la couleur d'un pixel, il suffit de passer les coordonnées x et y et la valeur colorimétrique à l'une de ces méthodes.

Dans l'exemple suivant, `setPixel()` est utilisée pour tracer une croix sur un fond vert BitmapData. La méthode `getPixel()` permet ensuite de récupérer la valeur colorimétrique du pixel ayant les coordonnées 50, 50 et de suivre la valeur renvoyée.

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 100, false, 0x009900);

for (var i:uint = 0; i < 100; i++)
{
    var red:uint = 0xFF0000;
    myBitmapData.setPixel(50, i, red);
    myBitmapData.setPixel(i, 50, red);
}

var myBitmapImage:Bitmap = new Bitmap(myBitmapData);
addChild(myBitmapImage);

var pixelValue:uint = myBitmapData.getPixel(50, 50);
trace(pixelValue.toString(16));
```

Pour lire la valeur d'un groupe de pixels, et non pas d'un pixel isolé, utilisez la méthode `getPixels()`. Cette méthode génère un tableau d'octets à partir d'une zone rectangulaire de données de pixels, et le passe en paramètre. Chaque élément du tableau d'octets (autrement dit, les valeurs des pixels) est un entier non signé (valeurs non multipliées sur 32 bits).

Inversement, pour modifier (ou définir) la valeur d'un groupe de pixels, utilisez la méthode `setPixels()`. Cette méthode attend deux paramètres (`rect` et `inputByteArray`), qui sont combinés pour produire une zone rectangulaire (`rect`) de données de pixels (`inputByteArray`).

Au fur et à mesure de la lecture (ou de l'écriture) des données dans `inputByteArray`, la méthode `ByteArray.readUnsignedInt()` est appelée pour chaque pixel du tableau. Si pour une raison quelconque `inputByteArray` ne contient pas un rectangle complet de données de pixels, la méthode interrompt le traitement des données de l'image.

Il est important de comprendre que pour lire ou modifier des données de pixels, le tableau d'octets attend les valeurs suivantes sur 32 octets : alpha, rouge, vert, bleu (ARVB).

L'exemple suivant utilise les méthodes `getPixels()` et `setPixels()` pour copier un groupe de pixels d'un objet BitmapData à un autre :

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.utils.ByteArray;
import flash.geom.Rectangle;

var bitmapDataObject1:BitmapData = new BitmapData(100, 100, false, 0x006666FF);
var bitmapDataObject2:BitmapData = new BitmapData(100, 100, false, 0x00FF0000);

var rect:Rectangle = new Rectangle(0, 0, 100, 100);
var bytes:ByteArray = bitmapDataObject1.getPixels(rect);

bytes.position = 0;
bitmapDataObject2.setPixels(rect, bytes);

var bitmapImage1:Bitmap = new Bitmap(bitmapDataObject1);
addChild(bitmapImage1);
var bitmapImage2:Bitmap = new Bitmap(bitmapDataObject2);
addChild(bitmapImage2);
bitmapImage2.x = 110;
```

Détection de collision au niveau des pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `BitmapData.hitTest()` effectue une détection de collision au niveau des pixels entre les données bitmap et celles d'un autre objet ou d'un point.

La méthode `BitmapData.hitTest()` accepte cinq paramètres :

- `firstPoint` (Point) : ce paramètre référence la position du coin supérieur gauche du premier objet BitmapData sur lequel le test de collision doit être effectué.
- `firstAlphaThreshold` (uint) : ce paramètre spécifie la valeur la plus élevée du canal alpha considéré comme étant opaque pour ce test de collision.
- `secondObject` (Object) : ce paramètre représente la zone d'impact. L'objet `secondObject` peut être un objet de type Rectangle, Point, Bitmap ou BitmapData. Cet objet représente la zone dans laquelle doit être effectuée la détection de collision.
- `secondBitmapDataPoint` (Point) : ce paramètre facultatif définit l'emplacement d'un pixel dans le deuxième objet BitmapData. Utilisez uniquement ce paramètre lorsque la valeur de `secondObject` est un objet BitmapData. La valeur par défaut est `null`.

- `secondAlphaThreshold` (uint) : ce paramètre facultatif représente la valeur la plus élevée de canal alpha considérée comme étant opaque dans le deuxième objet BitmapData. La valeur par défaut est 1. Utilisez uniquement ce paramètre lorsque la valeur de `secondObject` est un objet BitmapData et que les deux objets BitmapData sont transparents.

Lors d'une détection de collision sur des images opaques, n'oubliez pas qu'ActionScript traite l'image comme si c'était un rectangle entièrement opaque (ou un cadre de sélection). Par ailleurs, lors de tests de collision au niveau des pixels pour des images qui sont transparentes, les deux images doivent être transparentes. De plus, ActionScript utilise les paramètres de seuil alpha pour déterminer le point auquel les pixels passent de transparents à opaques.

L'exemple suivant crée trois images bitmap et teste une éventuelle collision de pixels en deux différents points (l'un renvoie false, l'autre true) :

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.geom.Point;

var bmd1:BitmapData = new BitmapData(100, 100, false, 0x000000FF);
var bmd2:BitmapData = new BitmapData(20, 20, false, 0x00FF3300);

var bm1:Bitmap = new Bitmap(bmd1);
this.addChild(bm1);

// Create a red square.
var redSquare1:Bitmap = new Bitmap(bmd2);
this.addChild(redSquare1);
redSquare1.x = 0;

// Create a second red square.
var redSquare2:Bitmap = new Bitmap(bmd2);
this.addChild(redSquare2);
redSquare2.x = 150;
redSquare2.y = 150;

// Define the point at the top-left corner of the bitmap.
var pt1:Point = new Point(0, 0);
// Define the point at the center of redSquare1.
var pt2:Point = new Point(20, 20);
// Define the point at the center of redSquare2.
var pt3:Point = new Point(160, 160);

trace(bmd1.hitTest(pt1, 0xFF, pt2)); // true
trace(bmd1.hitTest(pt1, 0xFF, pt3)); // false
```

Copie de données bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous disposez de plusieurs méthodes pour copier les données bitmap d'une image à une autre : `clone()`, `copyPixels()`, `copyChannel()`, `draw()` et `drawWithQuality()` (la méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures).

Comme son nom l'indique, la méthode `clone()` permet de cloner, ou échantillonner, des données bitmap d'un objet BitmapData à un autre. Cette méthode renvoie un nouvel objet BitmapData qui est un clone exact de l'occurrence originale.

L'exemple suivant clone une copie d'un carré orange (parent) et place le clone à côté du carré parent original :

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myParentSquareBitmap:BitmapData = new BitmapData(100, 100, false, 0x00ff3300);
var myClonedChild:BitmapData = myParentSquareBitmap.clone();

var myParentSquareContainer:Bitmap = new Bitmap(myParentSquareBitmap);
this.addChild(myParentSquareContainer);

var myClonedChildContainer:Bitmap = new Bitmap(myClonedChild);
this.addChild(myClonedChildContainer);
myClonedChildContainer.x = 110;
```

La méthode `copyPixels()` permet de copier rapidement et aisément des pixels d'un objet BitmapData dans un autre. Cette méthode prend un « cliché » rectangulaire (défini par le paramètre `sourceRect`) de l'image source et le copie dans une autre zone rectangulaire de taille égale. L'emplacement du rectangle ainsi « collé » est défini par le paramètre `destPoint`.

La méthode `copyChannel()` analyse une valeur de canal de couleur prédéfini (alpha, rouge, vert ou bleu) dans un objet source BitmapData et la copie dans un canal donné de l'objet BitmapData de destination. Cette méthode n'affecte pas les autres canaux de l'objet BitmapData de destination.

Les méthodes `draw()` et `drawWithQuality()` dessinent ou affichent le contenu graphique d'un objet d'affichage source (Sprite, Clip, etc.) dans un nouveau bitmap. Les paramètres `matrix`, `colorTransform`, `blendMode` et `clipRect` permettent de modifier l'aspect du nouveau bitmap. Cette méthode utilise le programme de rendu vectoriel de Flash Player et AIR pour générer les données.

Pour appeler la méthode `draw()` ou `drawWithQuality()`, vous devez lui transmettre l'objet source (Sprite, Clip ou tout autre objet d'affichage) comme premier paramètre, comme ci-dessous :

```
myBitmap.draw(movieClip);
```

Si des transformations (couleur, matrice, etc.) ont été appliquées à l'objet source après son chargement, ces transformations ne sont pas copiées dans le nouvel objet. Pour copier les transformations dans le nouveau bitmap, vous devez copier la valeur de la propriété `transform` de l'objet original dans la propriété `transform` de l'objet Bitmap qui utilise le nouvel objet BitmapData.

Compression des données d'une image bitmap

Flash Player 11.3 et les versions ultérieures, AIR 3.3 et les versions ultérieures

La méthode `flash.display.BitmapData.encode()` permet de compresser de façon native les données d'une image bitmap dans l'un des formats de compression d'image suivants :

- **PNG** : utilise la compression PNG, en ayant éventuellement recours à une compression rapide pour augmente la vitesse de compression selon la taille du fichier. Pour utiliser la compression PNG, transmettez un nouvel objet `flash.display.PNGEncoderOptions` comme second paramètre de la méthode `BitmapData.encode()`.

- **JPEG** : utilise la compression JPEG, en spécifiant éventuellement la qualité de l'image. Pour utiliser la compression JPEG, transmettez un nouvel objet `flash.display.JPEGEncoderOptions` comme second paramètre de la méthode `BitmapData.encode()`.
- **JPEGXR** : utilise la compression JPEG Extended Range (XR), en spécifiant éventuellement les paramètres Canal de couleur, Avec perte et Entropie. Pour utiliser la compression JPEGXR, transmettez un nouvel objet `flash.display.JPEGXREncoderOptions` comme second paramètre de la méthode `BitmapData.encode()`.

Vous pouvez utiliser cette fonction pour le traitement des images dans le cadre d'un flux de chargement ou de téléchargement sur le serveur.

L'exemple de fragment de code suivant compresse l'objet BitmapData avec `JPEGEncoderOptions` :

```
// Compress a BitmapData object as a JPEG file.
var bitmapData:BitmapData = new BitmapData(640,480,false,0x00FF00);
var byteArray:ByteArray = new ByteArray();
bitmapData.encode(new Rectangle(0,0,640,480), new flash.display.JPEGEncoderOptions(),
byteArray);
```

Création de textures avec les fonctions de bruit aléatoire

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour modifier l'aspect d'un bitmap, vous pouvez lui appliquer un effet de bruit à l'aide de la méthode `noise()` ou de la méthode `perlinNoise()`. La « neige » qui apparaît sur l'écran d'un téléviseur mal réglé est du bruit, ou du souffle.

Pour appliquer un effet de bruit à un bitmap, utilisez la méthode `noise()`. Cette méthode applique une valeur colorimétrique aléatoire aux pixels de la zone spécifiée d'une image bitmap.

Cette méthode accepte cinq paramètres :

- `randomSeed` (int) : valeur aléatoire de départ qui déterminera le motif. En dépit du nom de ce paramètre, une même valeur pour ce nombre produira toujours le même résultat. Pour obtenir un résultat véritablement aléatoire, utilisez la méthode `Math.random()` pour transmettre un nombre aléatoire pour ce paramètre.
- `low` (uint) : ce paramètre représente la valeur la plus faible à générer pour chaque pixel (de 0 à 255). Sa valeur par défaut est 0. Les valeurs les plus basses produisent un motif de bruit sombre, les valeurs les plus élevées produisent un motif de plus en plus clair.
- `high` (uint) : ce paramètre représente la valeur la plus élevée à générer pour chaque pixel (de 0 à 255). La valeur par défaut est 255. Les valeurs les plus basses produisent un motif de bruit sombre, les valeurs les plus élevées produisent un motif de plus en plus clair.
- `channelOptions` (uint) : ce paramètre indique le canal de couleur de l'objet bitmap auquel le motif de bruit sera appliqué. Ce nombre peut être une combinaison quelconque des quatre valeurs des canaux de couleur (ARVB). La valeur par défaut est 7.
- `grayScale` (Boolean) : lorsque ce paramètre est activé (`true`), la valeur de `randomSeed` est appliquée aux pixels du bitmap, ce qui donne concrètement un effet de délavé sur toutes les couleurs de l'image. Le canal alpha n'est pas affecté par ce paramètre. La valeur par défaut est `false`.

L'exemple suivant crée une image bitmap et lui applique un motif de bruit bleu :

```
package
{
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.BitmapDataChannel;

    public class BitmapNoise1 extends Sprite
    {
        public function BitmapNoise1()
        {
            var myBitmap:BitmapData = new BitmapData(250, 250,false, 0xff000000);
            myBitmap.noise(500, 0, 255, BitmapDataChannel.BLUE,false);
            var image:Bitmap = new Bitmap(myBitmap);
            addChild(image);
        }
    }
}
```

Si vous souhaitez créer une texture d'aspect plus organique, utilisez la méthode `perlinNoise()`. La méthode `perlinNoise()` produit des textures organiques plus réalistes, qui sont idéales pour des effets de fumée, de nuage, d'eau, de flamme ou même d'explosion.

Comme la méthode `perlinNoise()` fait appel à un algorithme, elle nécessite moins de mémoire vive que les textures à base de bitmaps. Elle peut malgré tout consommer beaucoup de ressources processeur, ce qui risque de ralentir le contenu et de provoquer des actualisations d'écran plus lentes que la cadence nominale, en particulier sur les ordinateurs plus anciens. En effet, les algorithmes de cette méthode effectuent des calculs en virgule flottante.

Cette méthode accepte neuf paramètres (les six premiers sont obligatoires) :

- `baseX` (Number) : détermine la valeur x (taille) des motifs créés.
- `baseY` (Number) : détermine la valeur y (taille) des motifs créés.
- `numOctaves:` (uint) : nombre d'octaves ou fonctions de bruit individuelles à combiner pour créer ce bruit. Les nombres d'octaves élevés offrent davantage de détails mais nécessitent également un temps de traitement plus important.
- `randomSeed` (int) : la valeur aléatoire de départ fonctionne exactement comme dans la fonction `noise()`. Pour obtenir un résultat véritablement aléatoire, utilisez la méthode `Math.random()` pour transmettre un nombre aléatoire pour ce paramètre.
- `stitch` (Boolean) : si la valeur est `true`, cette méthode tente de lisser les bords de transition de l'image pour créer des textures sans bords définis, en vue d'une utilisation en mosaïque.
- `fractalNoise` (Boolean) : ce paramètre concerne les bords des dégradés générés par cette méthode. S'il a la valeur `true`, la méthode génère un bruit fractal qui adoucit le pourtour. S'il a la valeur `false`, la méthode génère des turbulences. Les dégradés d'une image créée avec des turbulences présentent des discontinuités visibles qui permettent de mieux restituer certains effets visuels comme les flammes ou les vagues.
- `channelOptions` (uint) : le paramètre `channelOptions` fonctionne exactement comme dans la méthode `noise()`. Il indique le canal de couleur de l'objet bitmap auquel le motif de bruit sera appliqué. Ce nombre peut être une combinaison quelconque des quatre valeurs des canaux de couleur (ARVB). La valeur par défaut est 7.
- `grayScale` (Boolean) : le paramètre `grayScale` fonctionne exactement comme dans la méthode `noise()`. Si ce paramètre est activé (`true`), la valeur de `randomSeed` est appliquée aux pixels du bitmap, ce qui donne concrètement un effet de délavé sur toutes les couleurs de l'image. La valeur par défaut est `false`.

- `offsets`: (Array) : un tableau de points correspondant aux décalages x et y pour chaque octave. En modifiant les valeurs de décalage, vous pouvez faire défiler en continu les calques d'une image. Chaque point du tableau de décalage affecte une fonction de bruit pour une octave spécifique. La valeur par défaut est `null`.

L'exemple suivant crée un objet BitmapData de 150 x 150 pixels qui appelle la méthode `perlinNoise()` pour générer un effet de nuage vert et bleu :

```
package
{
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.BitmapDataChannel;

    public class BitmapNoise2 extends Sprite
    {
        public function BitmapNoise2()
        {
            var myBitmapDataObject:BitmapData =
                new BitmapData(150, 150, false, 0x00FF0000);

            var seed:Number = Math.floor(Math.random() * 100);
            var channels:uint = BitmapDataChannel.GREEN | BitmapDataChannel.BLUE
            myBitmapDataObject.perlinNoise(100, 80, 6, seed, false, true, channels, false, null);

            var myBitmap:Bitmap = new Bitmap(myBitmapDataObject);
            addChild(myBitmap);
        }
    }
}
```

Défilement du contenu d'images bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Supposons que vous avez créé une application cartographique. Chaque fois que l'utilisateur déplace la carte, vous devez actualiser l'affichage (même si le plan n'a été déplacé que de quelques pixels).

Pour obtenir cette fonctionnalité, il est possible de réafficher une nouvelle image contenant le plan actualisé à chaque déplacement. Il est également possible de créer une grande image globale et d'utiliser la méthode `scroll()`.

La méthode `scroll()` copie une image bitmap affichée et la colle à un nouvel emplacement, spécifié par les paramètres (`x`, `y`). S'il se trouve qu'une partie de l'image est située hors écran, l'effet obtenu est celui d'un défilement de l'image. Si cette fonction est combinée avec un timer (ou un événement `enterFrame`), l'image semble animée.

L'exemple suivant reprend l'exemple précédent et génère une image bitmap de grande taille (dont les trois-quarts sont restitués hors scène). La méthode `scroll()` est alors appliquée. Grâce à un écouteur pour l'événement `enterFrame`, l'image est décalée d'un pixel en diagonale vers le bas. Cette méthode est appelée pour chaque nouvelle image. Les parties de l'image non affichées initialement apparaissent alors peu à peu sur la scène grâce à ce défilement.

```
import flash.display.Bitmap;
import flash.display.BitmapData;

var myBitmapDataObject:BitmapData = new BitmapData(1000, 1000, false, 0x00FF0000);
var seed:Number = Math.floor(Math.random() * 100);
var channels:uint = BitmapDataChannel.GREEN | BitmapDataChannel.BLUE;
myBitmapDataObject.perlinNoise(100, 80, 6, seed, false, true, channels, false, null);

var myBitmap:Bitmap = new Bitmap(myBitmapDataObject);
myBitmap.x = -750;
myBitmap.y = -750;
addChild(myBitmap);

addEventListener(Event.ENTER_FRAME, scrollBitmap);

function scrollBitmap(event:Event):void
{
    myBitmapDataObject.scroll(1, 1);
}
```

Utilisation du mipmapping

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les *mipmaps* sont des images bitmap qui sont regroupées et associées à une texture dans le but d'améliorer la qualité et les performances d'affichage à l'exécution. Chaque image bitmap dans le mipmap est une version de l'image bitmap principale, mais à un niveau de détails réduit.

Par exemple, vous pouvez disposer d'un mipmap qui inclut une image principale à une qualité optimale de 64 × 64 pixels. Les images de qualité inférieure dans le mipmap seront de 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2 et de 1 × 1 pixels.

La *diffusion en continu de texture* fait référence à la capacité de charger tout d'abord les images bitmap de qualité inférieure, puis d'afficher progressivement les images bitmap de qualité supérieure au fur et à mesure de leur chargement. Etant donné que les bitmaps de qualité inférieure sont de petites images, elles se chargent plus rapidement que l'image principale. Par conséquent, les utilisateurs de l'application peuvent afficher l'image dans une application avant le chargement de l'image bitmap principale de qualité supérieure.

Flash Player 9.115.0 (et les versions ultérieures) et AIR implémentent cette technique (dite de *mip-mapping*), en créant des versions optimisées à diverses échelles de chaque bitmap (en partant de 50 %).

Flash Player 11.3 et AIR 3.3 prennent en charge la diffusion en continu de textures via le paramètre `streamingLevels` des méthodes `Context3D.createCubeTexture()` et `Context3D.createTexture()`.

La compression de texture permet d'enregistrer des images de texture au format compressé directement sur le GPU en vue d'économiser la mémoire GPU et la bande passante. En règle générale, les textures sont compressées hors ligne et téléchargées sur le GPU au format compressé. Néanmoins, Flash Player 11.4 et AIR 3.4 prennent en charge la compression de texture à l'exécution, utile dans certaines situations, notamment lors du rendu des textures dynamiques d'une illustration vectorielle. Pour utiliser la compression de texture à l'exécution, procédez comme suit :

- Créez l'objet de texture en appelant la méthode `Context3D.createTexture()`, et en transmettant `flash.display3D.Context3DTextureFormat.COMPRESSED` ou `flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA` dans le troisième paramètre.

- A l'aide de l'occurrence `flash.display3D.textures.Texture` renvoyée par `createTexture()`, appelez la méthode `flash.display3D.textures.Texture.uploadFromBitmapData()` ou `flash.display3D.textures.Texture.uploadFromByteArray()`. Ces méthodes permettent de télécharger et de compresser simultanément la texture.

Les mipmaps sont créées pour les types de bitmap suivants :

- Une image bitmap (fichiers JPEG, GIF ou PNG) affichée par le biais de la classe Loader d'ActionScript 3.0.
- Une image bitmap dans la bibliothèque d'un document Flash Professional.
- Un objet BitmapData.
- Une image bitmap affichée à l'aide de la fonction `loadMovie()` d'ActionScript 2.0.

Les mipmaps ne sont pas appliqués aux objets filtrés ni aux clips dont les bitmaps sont en cache. En revanche, ils sont appliqués si un objet d'affichage filtré contient des transformations de bitmap, même si le bitmap se trouve dans un contenu masqué.

Le mipmapping est exécuté automatiquement, mais les quelques conseils suivants vous permettront d'être certain que vos images bénéficient de cette optimisation :

- Pour la lecture vidéo, définissez la propriété `smoothing` sur `true` pour l'objet Video (voir la classe Video).
- Pour les bitmaps, il n'est pas nécessaire de définir la propriété `smoothing` sur `true`, mais l'activation de cette propriété assure une amélioration visible de la qualité.
- Utilisez des tailles de bitmap divisibles par 4 ou 8 pour les images bidimensionnelles (affichage 640 x 128, qui peut être réduit comme suit : 320 x 64 > 160 x 32 > 80 x 16 > 40 x 8 > 20 x 4 > 10 x 2 > 5 x 1).

  Pour les textures tridimensionnelles, utilisez des mipmaps dans lesquels la résolution de chaque image est une puissance de 2 (2^n). Par exemple, la résolution de l'image principale est de 1 024 x 1 024 pixels. Les images de qualité inférieure dans le mipmap seront à 512 x 512, 256 x 256, 128 x 128 jusqu'à 1 x 1 pixels pour un total de 11 images dans le mipmap.

  Notez que le mip-mapping ne gère pas un contenu de bitmap dont la largeur ou la hauteur est impaire.

Exemple d'objet Bitmap : lune en rotation animée

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple de lune en rotation animée illustre les techniques d'utilisation des objets Bitmap et des données d'image bitmap (objets BitmapData). L'exemple crée une animation d'une lune sphérique en rotation et utilise comme données d'image brutes une image plane de la surface de la lune. Les techniques suivantes sont illustrées :

- Chargement d'une image externe et accès aux données d'image brutes correspondantes
- Création d'une animation par copie répétée des pixels de différentes parties d'une image source
- Création d'une image bitmap par définition de la valeur des pixels

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application de la lune en rotation animée résident dans le dossier Samples/SpinningMoon. L'application se compose des fichiers suivants :

Fichier	Description
SpinningMoon.mxml ou SpinningMoon.fla	Le fichier d'application principal dans Flex (MXML) ou Flash (FLA).
com/example/programmingas3/moon/MoonSphere.as	Classe exécutant le chargement, l'affichage et l'animation de la lune.
moonMap.png	Fichier image contenant une photographie de la surface de la lune, chargé et utilisé pour créer la lune en rotation animée.

Chargement d'une image externe comme données bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La première tâche à exécuter dans cet exemple consiste à charger une image externe, une photographie de la surface de la lune. Le chargement est géré par deux méthodes de la classe MoonSphere : le constructeur `MoonSphere()`, qui lance le processus de chargement, et la méthode `imageLoadComplete()`, qui est appelée au terme du chargement de l'image externe.

Le chargement d'une image externe est similaire à celui d'un fichier SWF externe : il est réalisé par une occurrence de la classe flash.display.Loader. Le code ci-après de la méthode `MoonSphere()` commence à charger l'image :

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

La première ligne déclare l'occurrence de Loader appelée `imageLoader`. La troisième ligne commence le processus de chargement à proprement parler en appelant la méthode `load()` de l'objet Loader. Cette méthode transmet une occurrence URLRequest représentant l'URL de l'image à charger. La deuxième ligne définit l'écouteur d'événement qui se déclenchera à l'issue du chargement de l'image. Observez que la méthode `addEventListener()` n'est pas appelée sur l'occurrence de Loader elle-même, mais sur la propriété `contentLoaderInfo` de l'objet Loader. L'occurrence de Loader n'envoie pas d'événements en rapport avec le contenu chargé. En revanche, sa propriété `contentLoaderInfo` contient une référence à l'objet LoaderInfo qui est associé au contenu chargé dans l'objet Loader (en l'occurrence, l'image externe). L'objet LoaderInfo génère des événements en rapport avec le déroulement et la fin du chargement du contenu externe, notamment l'événement `complete` (`Event.COMPLETE`) qui déclenche un appel à la méthode `imageLoadComplete()` au terme du chargement de l'image.

S'il est essentiel de lancer le chargement de l'image externe, il est tout aussi important de savoir comment procéder au terme de cette opération. Comme l'illustre le code ci-dessus, la fonction `imageLoadComplete()` est appelée une fois l'image chargée. Cette fonction exécute diverses opérations sur les données chargées, comme indiqué ultérieurement. Cependant, pour utiliser les données d'image, elle doit pouvoir y accéder. Une image externe chargée par le biais d'un objet Loader devient une image Bitmap jointe en tant qu'objet d'affichage enfant de l'objet Loader. Dans ce cas, la méthode écouteur d'événement a accès à l'occurrence de Loader dans l'objet événement transmis en tant que paramètre à la méthode. Les premières lignes de la méthode `imageLoadComplete()` sont les suivantes :

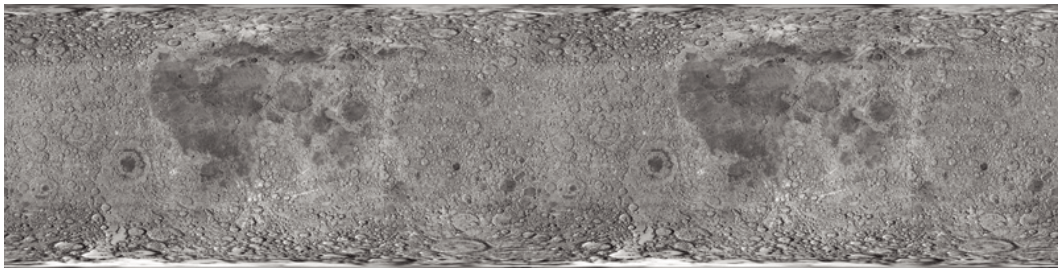
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

Notez que le paramètre de l'objet événement s'appelle `event` et que c'est une occurrence de la classe Event. Chaque occurrence de la classe Event possède une propriété `target`, qui fait référence à l'objet déclenchant l'événement (en l'occurrence, l'occurrence de LoaderInfo sur laquelle la méthode `addEventListener()` a été appelée, comme indiqué plus haut). De même, l'objet LoaderInfo possède une propriété `content` qui, à l'issue du chargement, contient une occurrence de Bitmap comportant l'image bitmap chargée. Pour afficher l'image directement à l'écran, vous pouvez joindre cette occurrence de Bitmap (`event.target.content`) à un conteneur d'objet d'affichage (vous pouvez aussi joindre l'objet Loader à un conteneur d'objet d'affichage). Toutefois, dans cet exemple, le contenu chargé n'est pas affiché à l'écran, il est utilisé en tant que données d'image brutes. Par conséquent, la première ligne de la méthode `imageLoadComplete()` lit la propriété `bitmapData` de l'occurrence de Bitmap chargée (`event.target.content.bitmapData`) et la stocke dans la variable d'occurrence de `textureMap`, qui est utilisée en tant que source des données d'image pour créer l'animation de la lune en rotation. Ce processus est décrit ci-après.

Création d'une animation par copie de pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une animation, dans sa définition la plus simple, est l'illusion d'un mouvement ou d'un changement, créée par la modification graduelle d'une image. Cet exemple a pour but de créer l'illusion d'une lune sphérique tournant sur son axe vertical. Cependant, pour les besoins de l'animation, vous pouvez ne pas tenir compte de l'aspect de distorsion sphérique de l'exemple. Examinez l'image chargée et utilisée comme source des données d'image de la lune :



Comme vous pouvez le constater, l'image ne représente pas une ou plusieurs sphères ; c'est une photographie rectangulaire de la surface de la lune. La photographie ayant été prise à l'emplacement exact de l'équateur de la lune, les parties supérieures et inférieures de l'image sont donc étirées et déformées. Pour supprimer la distorsion de l'image et lui redonner son aspect sphérique, nous utiliserons un filtre Mappage de déplacement (voir plus bas). Toutefois, l'image source étant un rectangle, il suffit que le code fasse glisser horizontalement la photographie de la surface de la lune pour créer l'illusion d'une sphère en rotation.

Observez que l'image contient en fait deux copies juxtaposées de la photographie de la surface de la lune. Cette image représente l'image source dans laquelle des données ont été copiées plusieurs fois pour créer un effet de mouvement. La juxtaposition de deux copies de l'image facilite la création d'un effet de défilement continu. Examinons en détail le processus d'animation afin de mieux le comprendre.

Le processus s'applique à deux objets ActionScript distincts. Le premier de ces objets est l'image source chargée qui, dans le code, est représentée par l'occurrence de BitmapData `textureMap` Comme nous l'avons vu, les données d'image sont insérées dans `textureMap` dès le chargement de l'image externe à l'aide de ce code :

```
textureMap = event.target.content.bitmapData;
```

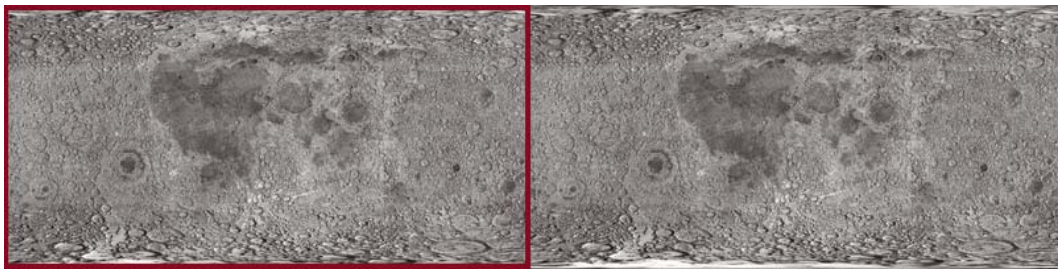
Le contenu de `textureMap` correspond à l'image de la lune rectangulaire. En outre, pour créer la rotation animée, le code utilise l'occurrence de Bitmap `sphere`, qui représente l'objet d'affichage qui affiche l'image de la lune à l'écran. A l'instar de `textureMap`, l'objet `sphere` contient les données d'image initiales de la méthode `imageLoadComplete()`, ainsi que le stipule le code suivant :

```
sphere = new Bitmap();
sphere.bitmapData = new BitmapData(textureMap.width / 2, textureMap.height);
sphere.bitmapData.copyPixels(textureMap,
                        new Rectangle(0, 0, sphere.width, sphere.height),
                        new Point(0, 0));
```

Comme vous pouvez le constater, `sphere` est instancié. La hauteur et la largeur de sa propriété `bitmapData` (les données d'image brutes qui sont affichées par `sphere`) sont identiques à celles de `textureMap`. Autrement dit, le contenu de `sphere` a la même taille qu'une seule photographie de la lune (puisque l'image `textureMap` contient deux photographies juxtaposées). Des données d'image sont ensuite insérées dans la propriété `bitmapData` à l'aide de sa méthode `copyPixels()`. Les paramètres de l'appel de la méthode `copyPixels()` donnent plusieurs indications :

- Le premier paramètre indique que les données d'image copiées proviennent de `textureMap`.
- Le deuxième paramètre, une nouvelle occurrence de Rectangle, détermine quelle partie de `textureMap` est copiée. En l'occurrence, le cliché est un rectangle dont l'origine coïncide avec le coin supérieur gauche de `textureMap` (ce qu'indiquent les deux premiers paramètres de `Rectangle()` : `0, 0`) et dont la largeur et la hauteur correspondent aux propriétés `width` et `height` de `sphere`.
- Le troisième paramètre, une nouvelle occurrence de Point avec des valeurs de x et y égales à `0`, définit la destination des données de pixel ; en l'occurrence, le coin supérieur gauche (0, 0) de `sphere.bitmapData`.

Représenté visuellement, le code copie les pixels de `textureMap` mis en évidence ci-dessous et les colle sur `sphere`. Autrement dit le contenu BitmapData de `sphere` correspond à la partie de `textureMap` mise en évidence :



Pour rappel, il s'agit seulement de l'état initial de `sphere`, le contenu de la première image copiée sur `sphere`.

Une fois l'image source chargée et `sphere` créé, il ne reste plus à la méthode `imageLoadComplete()` qu'à définir l'animation. L'animation est pilotée par une occurrence de Timer, `rotationTimer`, créée et lancée par le code suivant :

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

Le code commence par créer l'occurrence de Timer `rotationTimer`. Le paramètre passé au constructeur `Timer()` indique que `rotationTimer` doit déclencher son événement `timer` toutes les 15 millisecondes. La méthode `addEventListener()` qui est appelée ensuite stipule que le déclenchement de l'événement `timer` (`TimerEvent.TIMER`) entraîne l'appel de la méthode `rotateMoon()`. Enfin, l'appel de la méthode `start()` du timer entraîne le démarrage de celui-ci.

De par la définition de `rotationTimer`, Flash Player appelle la méthode `rotateMoon()` dans la classe MoonSphere environ toutes les 15 millisecondes, ce qui se traduit par l'animation de la lune. Le code source de la méthode `rotateMoon()` est le suivant :

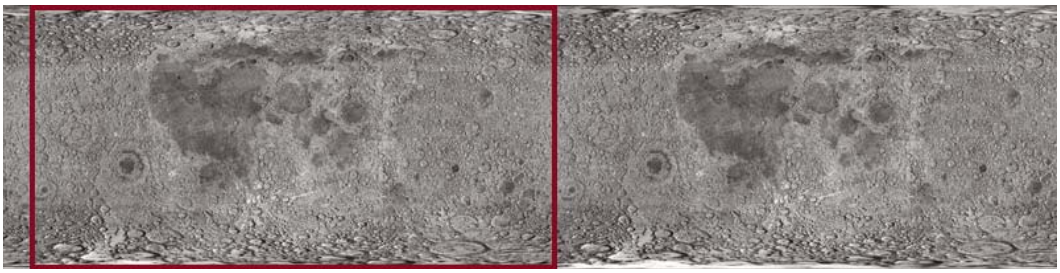
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                    new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                    new Point(0, 0));

    event.updateAfterEvent();
}
```

Ce code effectue trois opérations :

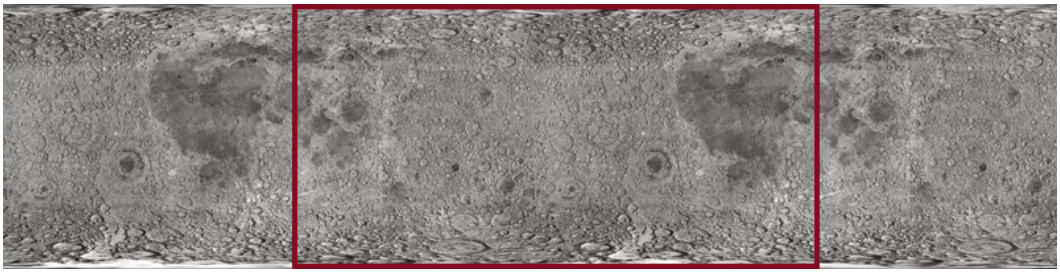
1 La valeur de la variable `sourceX` (initialement fixée à 0) est incrémentée d'une unité.

```
sourceX += 1;
```

`sourceX` permet de déterminer d'où proviennent, dans `textureMap`, les pixels copiés sur `sphere`. Ce code déplace donc le rectangle d'un pixel vers la droite sur `textureMap`. Comme le montre l'illustration suivante, après plusieurs cycles d'animation, le rectangle source s'est déplacé de plusieurs pixels vers la droite :

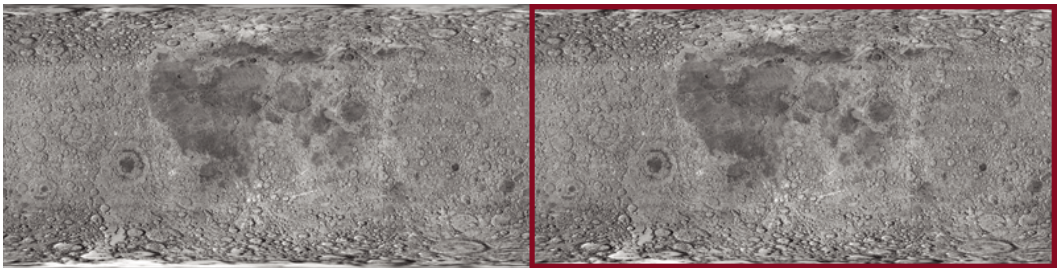


Après plusieurs autres cycles, le rectangle se trouve encore plus à droite.



C'est sur ce déplacement progressif constant de l'emplacement d'origine des pixels copiés que repose l'animation. Par un déplacement lent mais continu de l'emplacement source vers la droite, l'image affichée à l'écran dans `sphere` semble continuellement glisser vers la gauche. C'est pourquoi l'image source (`textureMap`) doit contenir deux copies de la photographie de la surface de la lune. Comme le rectangle se déplace continuellement vers la droite, il chevauche généralement les deux photographies et non pas seulement l'une d'elles.

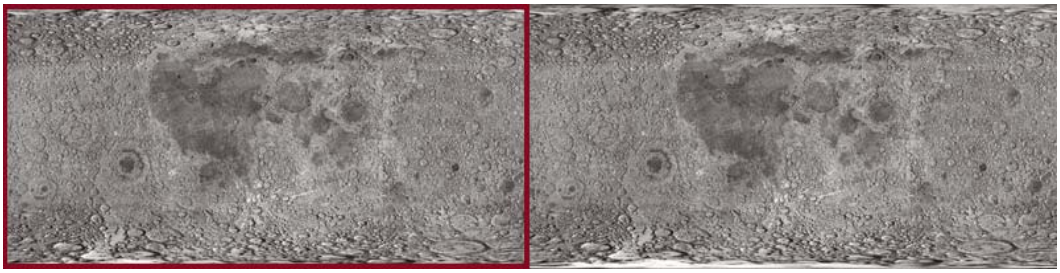
2 Ce lent déplacement vers la droite donne cependant lieu à un problème. Le rectangle finira par atteindre le bord droit de `textureMap` et ne trouvera plus de pixels à copier sur `sphere` :



Les lignes suivantes du code permettent de résoudre ce problème :

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

Le code vérifie si `sourceX` (le bord gauche du rectangle) a atteint le milieu de `textureMap`. Si tel est le cas, il remet la variable `sourceX` à 0 ; autrement dit, il la ramène au bord gauche de `textureMap`, et le cycle recommence :



3 Une fois la valeur de `sourceX` appropriée calculée, la dernière étape du processus d'animation consiste à copier les pixels du nouveau rectangle source sur `sphere`. Pour ce faire, nous reprenons le code qui a initialement rempli `sphere` (voir plus haut), à la différence près que, dans l'appel du constructeur `new Rectangle()`, le bord gauche du rectangle est placé à `sourceX` :

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

Pour rappel, ce code est appelé toutes les 15 millisecondes. Comment l'emplacement du rectangle source change constamment et que les pixels sont copiés sur `sphere`, à l'écran, la photographie de la lune représentée par `sphere` semble glisser continuellement. En d'autres termes, la lune semble tourner sur elle-même continuellement.

Définition de l'aspect sphérique

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La lune est bien entendu sphérique , ce n'est pas un rectangle. La photographie rectangulaire de la surface lunaire, qui fait l'objet d'une animation constante, doit donc être convertie en sphère. Cette opération comprend deux étapes : un masque cache tout le contenu excepté une partie circulaire de la photographie et un filtre Mappage de déplacement déforme l'apparence de la photographie, lui donnant un aspect tridimensionnel.

Dans un premier temps, un masque circulaire cache entièrement le contenu de l'objet MoonSphere excepté la sphère créée par le filtre. Le code suivant crée le masque, une occurrence de Shape, et l'applique à l'occurrence de MoonSphere :

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

Comme MoonSphere est un objet d'affichage (fondé sur la classe Sprite), il est possible d'appliquer directement le masque à l'occurrence de MoonSphere à l'aide de sa propriété `mask` héritée.

Il ne suffit pas d'occulter des parties de la photographie à l'aide d'un masque circulaire pour créer un effet réaliste de sphère en rotation. En raison de la façon dont la photographie de la surface lunaire a été prise, ses dimensions ne sont pas proportionnelles . les parties supérieures et inférieures de l'image sont déformées et étirées par rapport aux zones équatoriales. Pour déformer l'apparence de la photographie et lui donner un aspect tridimensionnel, nous allons utiliser un filtre Mappage de déplacement.

Ce type de filtre permet de déformer une image. En l'occurrence, nous allons déformer la photographie de la lune pour lui donner un aspect plus réaliste, en compressant horizontalement les parties supérieures et inférieures de l'image sans toucher à son milieu. En supposant que le filtre intervienne sur une partie carrée de la photographie, la compression du haut et du bas mais pas du milieu aura pour effet de convertir le carré en cercle. L'animation de cette image déformée a un effet secondaire : la distance en pixels parcourue par le milieu de l'image semble supérieure à celle couverte par les parties supérieure et inférieure, d'où l'impression que le cercle est en fait un objet tridimensionnel (une sphère).

Le code suivant permet de créer un filtre Mappage de déplacement appelé `displaceFilter` :

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

Le premier paramètre, `fisheyeLens`, est l'image de mappage ; en l'occurrence, un objet BitmapData créé par programmation. La création de cette image est décrite dans « Création d'une image bitmap par définition de la valeur des pixels » à la page 270. Les autres paramètres décrivent l'emplacement d'application du filtre au sein de l'image filtrée, les canaux colorimétriques utilisés pour régir l'effet de déplacement et leur impact sur celui-ci. Une fois le filtre Mappage de déplacement créé, il est appliqué à `sphere`, toujours dans la méthode `imageLoadComplete()` :

```
sphere.filters = [displaceFilter];
```

L'image finale, une fois le masque et le filtre appliqués, se présente comme suit :

A chaque cycle du processus d'animation de la lune en rotation, le contenu BitmapData de sphere est remplacé par un nouveau cliché des données d'image source. Il est cependant inutile de réappliquer le filtre à chaque fois car il est appliqué à l'occurrence de Bitmap (l'objet d'affichage) plutôt qu'aux données bitmap (données de pixel brutes). Pour rappel, l'occurrence de Bitmap ne correspond pas aux données bitmap. C'est un objet d'affichage qui affiche ces données à l'écran. Une occurrence de Bitmap peut être assimilée à un projecteur de diapositives, tandis qu'un objet BitmapData serait une diapositive présentée par le biais du projecteur. Il est possible d'appliquer un filtre directement à un objet BitmapData, ce qui reviendrait à dessiner sur une diapositive pour modifier l'image. Vous pouvez aussi appliquer un filtre à tout objet d'affichage, y compris une occurrence de Bitmap, ce qui équivaudrait à placer un filtre devant l'objectif du projecteur pour déformer l'image à l'écran sans modifier la diapositive d'origine. Comme les données bitmap brutes sont accessibles par le biais de la propriété bitmapData d'une occurrence de Bitmap, rien n'empêche de leur appliquer directement le filtre. Dans ce cas, cependant, il est préférable d'appliquer directement le filtre à l'objet d'affichage Bitmap plutôt qu'aux données bitmap.

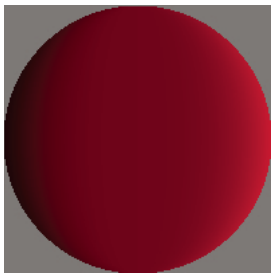
Pour plus d'informations sur l'utilisation du filtre Mappage de déplacement en ActionScript, voir « Filtrage des objets d'affichage » à la page 276.

Création d'une image bitmap par définition de la valeur des pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le fait qu'un filtre Mappage de déplacement implique en réalité deux images est un facteur important. L'image source est modifiée par le filtre. Dans cet exemple, il s'agit de l'occurrence de Bitmap `sphere`. L'autre image utilisée par le filtre est appelée l'image de mappage. Elle n'apparaît pas à l'écran. En revanche, la couleur de ses pixels est utilisée en entrée par la fonction de déplacement : la couleur d'un pixel se trouvant à des coordonnées x, y spécifiques détermine le déplacement (changement physique de position) à appliquer au pixel à ces coordonnées x, y dans l'image source.

Dans cet exemple, pour utiliser le filtre Mappage de déplacement en vue de créer un effet sphérique, il est donc nécessaire d'utiliser l'image de mappage appropriée, c'est-à-dire une image au fond gris comportant un cercle rempli d'un dégradé d'une seule couleur (rouge) qui passe, horizontalement, du foncé au clair, comme illustré ci-dessous :



Comme une image de mappage et un filtre uniques sont utilisés dans cet exemple, l'image de mappage est créée une seule fois, dans la méthode `imageLoadComplete()` (autrement dit, à l'issue du chargement de l'image externe). L'image de mappage, `fisheyeLens`, est créée par appel de la méthode `createFisheyeMap()` de la classe MoonSphere :

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

Au sein de la méthode `createFisheyeMap()`, l'image de mappage est dessinée pixel par pixel à l'aide de la méthode `setPixel()` de la classe BitmapData. Vous trouverez le code complet de la méthode `createFisheyeMap()` ci-dessous, suivi d'une présentation détaillée de son fonctionnement :

```
private function createFisheyeMap(radius:int):BitmapData
{
    var diameter:int = 2 * radius;

    var result:BitmapData = new BitmapData(diameter,
                                           diameter,
                                           false,
                                           0x808080);

    // Loop through the pixels in the image one by one
    for (var i:int = 0; i < diameter; i++)
    {
        for (var j:int = 0; j < diameter; j++)
        {
            // Calculate the x and y distances of this pixel from
            // the center of the circle (as a percentage of the radius).
            var pctX:Number = (i - radius) / radius;
            var pctY:Number = (j - radius) / radius;

            // Calculate the linear distance of this pixel from
            // the center of the circle (as a percentage of the radius).
            var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);

            // If the current pixel is inside the circle,
            // set its color.
            if (pctDistance < 1)
            {
                // Calculate the appropriate color depending on the
                // distance of this pixel from the center of the circle.
                var red:int;
                var green:int;
                var blue:int;
                var rgb:uint;
                red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
                green = 0;
                blue = 0;
                rgb = (red << 16 | green << 8 | blue);
                // Set the pixel to the calculated color.
                result.setPixel(i, j, rgb);
            }
        }
    }
    return result;
}
```

En premier lieu, la méthode reçoit un paramètre, `radius`, qui indique le rayon de l'image circulaire à créer. Le code crée ensuite l'objet BitmapData sur lequel sera tracé le cercle. Cet objet, appelé `result`, est renvoyé comme valeur résultante de la méthode. Comme illustré par l'extrait de code ci-dessous, la largeur et la hauteur de l'occurrence de BitmapData `result` créée sont égales au diamètre du cercle. En outre, cette occurrence n'a pas de transparence (le troisième paramètre correspond à `false`) et elle est pré-remplie par la couleur `0x808080` (gris moyen) :

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

Le code utilise ensuite deux boucles pour itérer par-dessus chaque pixel de l'image. La boucle extérieure parcourt de droite à gauche chaque colonne de l'image (la variable `i` représentant la position horizontale du pixel manipulé), alors que la boucle intérieure intervient sur chaque pixel de la colonne actuelle, de bas en haut (la variable `j` représentant la position verticale du pixel actuel). Le code des boucles (le contenu de la boucle intérieure étant omis) est illustré ci-dessous :

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

Au fur et à mesure de la manipulation des pixels par les boucles, une valeur est calculée à chacun d'eux (la valeur colorimétrique de ce pixel dans l'image de mappage). Ce processus comporte quatre étapes :

1. Le code calcule la distance séparant le pixel actuel du centre du cercle, le long de l'axe x (`i - radius`). Cette valeur est divisée par le rayon pour obtenir un pourcentage de celui-ci plutôt qu'une distance absolue (`(i - radius) / radius`). Ce pourcentage est stocké dans une variable appelée `pctX`. La valeur équivalente sur l'axe y est calculée et stockée dans la variable `pctY`, comme illustré dans le code ci-dessous :

   ```
   var pctX:Number = (i - radius) / radius;
   var pctY:Number = (j - radius) / radius;
   ```

2. Une formule trigonométrique standard (le théorème de Pythagore) est utilisée pour calculer la distance linéaire entre le centre du cercle et le point actuel, à partir de `pctX` et `pctY`. Cette valeur est stockée dans une variable, `pctDistance`, comme illustré ci-dessous :

   ```
   var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
   ```

3. Le code vérifie ensuite si la distance en pourcentage est inférieure à 1 (ou 100 % du rayon ; autrement dit, si le pixel concerné se trouve sur le rayon du cercle). Si le pixel figure dans le cercle, une valeur colorimétrique calculée lui est affectée (voir la description à l'étape 4). Dans le cas contraire, ce pixel ne fait l'objet d'aucune manipulation et conserve la couleur par défaut, c'est-à-dire le gris moyen.

   ```
   if (pctDistance < 1)
   {
       ...
   }
   ```

4. Une valeur colorimétrique est calculée pour tout pixel se trouvant dans le cercle. La couleur finale est une nuance de rouge qui va du noir (0 % de rouge) sur le bord gauche du cercle au rouge vif (100 % de rouge) sur le bord droit du cercle. La valeur colorimétrique comprend initialement trois parts de couleur (rouge, vert et bleu), comme illustré ci-dessous) :

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

Observez que seule la part rouge de la couleur (variable `red`) possède une valeur. Les valeurs vert et bleu (variables `green` et `blue`) sont illustrées ici par souci de clarté, mais il est possible de les omettre. Cette méthode ayant pour but de créer un cercle contenant un dégradé de rouge, les valeurs vertes et bleues sont superflues.

Une fois les trois valeurs colorimétriques individuelles déterminées, elles sont conjuguées dans une valeur entière unique à l'aide d'un algorithme de décalage de bits, comme illustré ci-dessous :

```
rgb = (red << 16 | green << 8 | blue);
```

En dernier lieu, la valeur colorimétrique calculée est affectée au pixel actuel à l'aide de la méthode `setPixel()` de l'objet BitmapData `result`, comme illustré ci-dessous :

```
result.setPixel(i, j, rgb);
```

Décodage asynchrone des images bitmap

Flash Player 11 et les versions ultérieures, Adobe AIR 2.6 et les versions ultérieures

Lorsque vous manipulez des images bitmap, vous pouvez décoder et charger ces dernières en mode asynchrone pour optimiser les performances perçues d'une application. Dans de nombreux cas, le décodage asynchrone d'une image bitmap prend autant de temps que le décodage synchrone. L'image bitmap est toutefois décodée dans un thread distinct avant que l'objet `Loader` associé n'envoie l'événement `COMPLETE`. Il est par conséquent possible de décoder en mode asynchrone des images de taille supérieure après leur chargement.

La classe `ImageDecodingPolicy` du package `flash.system` permet de stipuler le modèle de chargement de bitmap. Le modèle de chargement par défaut est synchrone.

Traitement du décodage de bitmap	Modèle de chargement de bitmap	Description
`ImageDecodingPolicy.ON_DEMAND`	Synchrone	Les images chargées sont décodées lorsque l'utilisateur accède aux données de l'image. Ce traitement est adapté au décodage des images de taille inférieure. Il s'avère également utile lorsque l'application ne nécessite pas d'effets et de transitions complexes.
`ImageDecodingPolicy.ON_LOAD`	Asynchrone	Les images chargées sont décodées au chargement, avant la distribution de l'événement `COMPLETE`. Ce mode est parfaitement adapté aux images de taille supérieure (supérieure à 10 MP). Si vous développez une application mobile AIR contenant des transitions de page, faites appel à ce traitement de chargement de bitmap pour optimiser les performances perçues de l'application.

Remarque : *si le fichier en cours de chargement est une image bitmap et que la méthode de décodage utilisée est* `ON_LOAD`*, l'image est décodée de façon asynchrone avant la distribution de l'événement* `COMPLETE`*.*

Le code suivant illustre l'utilisation de la classe `ImageDecodingPolicy` :

```
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_LOAD
var loader:Loader = new Loader();
loader.load(new URLRequest("http://www.adobe.com/myimage.png"), loaderContext);
```

Vous pouvez continuer à utiliser le traitement de décodage `ON_DEMAND` avec les méthodes `Loader.load()` et `Loader.loadBytes()`. Néanmoins, toutes les autres méthodes qui prennent un objet `LoaderContext` comme argument ignorent toutes les valeurs `ImageDecodingPolicy` transmises.

L'exemple suivant illustre la différence entre une image décodée en mode synchrone et en mode asynchrone :

```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.system.ImageDecodingPolicy;
    import flash.system.LoaderContext;

    public class AsyncTest extends Sprite
    {
        private var loaderContext:LoaderContext;
        private var loader:Loader;
        private var urlRequest:URLRequest;
        public function AsyncTest()
        {
            //Load the image synchronously
            loaderContext = new LoaderContext();
            //Default behavior.
            loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_DEMAND;
            loader = new Loader();
            loadImageSync();

            //Load the image asynchronously
            loaderContext = new LoaderContext();
            loaderContext.imageDecodingPolicy = ImageDecodingPolicy.ON_LOAD;
            loader = new Loader();
            loadImageASync();
        }

        private function loadImageASync():void{
            trace("Loading image asynchronously...");
            urlRequest = new URLRequest("http://www.adobe.com/myimage.png");
            urlRequest.useCache = false;
            loader.load(urlRequest, loaderContext);
            loader.contentLoaderInfo.addEventListener
                (Event.COMPLETE, onAsyncLoadComplete);
```

```
        }

        private function onAsyncLoadComplete(event:Event):void{
            trace("Async. Image Load Complete");
        }

        private function loadImageSync():void{
            trace("Loading image synchronously...");
            urlRequest = new URLRequest("http://www.adobe.com/myimage.png");
            urlRequest.useCache = false;
            loader.load(urlRequest, loaderContext);
            loader.contentLoaderInfo.addEventListener
                (Event.COMPLETE, onSyncLoadComplete);
        }

        private function onSyncLoadComplete(event:Event):void{
            trace("Sync. Image Load Complete");
        }
    }
}
```

Pour une démonstration de l'effet des diverses stratégies de décodage, voir Thibaud Imbert: Asynchronous bitmap decoding in the Adobe Flash runtimes (disponible en anglais uniquement).

Chapitre 14 : Filtrage des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Historiquement, l'application d'effets de filtres à des images bitmap est du domaine des logiciels spécialisés en retouche d'image, comme Adobe Photoshop® et Adobe Fireworks®. ActionScript 3.0 comprend le package flash.filters, qui intègre une série de classes de filtres d'effet d'image bitmap. Ces effets permettent aux développeurs d'appliquer des filtres aux images bitmap et objets d'affichage par le biais d'un programme afin d'obtenir une grande partie des effets disponibles dans les applications de retouche graphique.

Principes de base du filtrage des objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'une des façons de rendre une application plus séduisante est de lui ajouter des effets graphiques simples, comme une ombre portée derrière une photo pour créer l'illusion de la 3D, ou un rayonnement autour d'un bouton pour indiquer qu'il s'agit du bouton actif. ActionScript 3.0 comporte dix filtres qui peuvent être appliqués à n'importe quel objet d'affichage ou à une occurrence de BitmapData. Ces filtres vont des effets de base (filtres Ombre portée et Rayonnement) à des effets plus complexes, tels que le filtre Mappage du déplacement et le filtre Convolution matricielle.

Remarque : *outre les filtres intégrés, vous pouvez programmer des effets et filtres personnalisés par le biais de Pixel Bender (voir « Utilisation des shaders de Pixel Bender » à la page 310).*

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la création de filtres :

Biseau Effet de rebord créé en éclaircissant les pixels de deux côtés contigus et en assombrissant les pixels des deux côtés opposés, afin de créer un effet tridimensionnel de bordure fréquemment utilisé pour donner à des boutons et autres graphiques un effet enfoncé ou sorti.

Convolution Distorsion des pixels d'une image obtenue en combinant la valeur de chaque pixel avec celle(s) d'un ou plusieurs des pixels voisins, selon divers pourcentages.

Déplacement Décalage des pixels d'une image vers une nouvelle position.

Matrice Grille de chiffres utilisée pour effectuer certains calculs mathématiques, en appliquant ces chiffres à diverses valeurs et en combinant le résultat.

Voir aussi

Package flash.filters

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

Création et application de filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les filtres permettent d'appliquer divers effets (ombre portée, biseau, flou, etc.) à des images bitmap et à des objets d'affichage. Chaque filtre étant défini sous forme de classe, il suffit pour appliquer un filtre de créer une occurrence d'un objet filtre, ce qui n'est guère différent de la création de tout autre objet. Après avoir créé une occurrence d'un objet filtre, il est facile de l'appliquer à un objet d'affichage à l'aide de la propriété `filters` de cet objet ou, dans le cas d'un objet BitmapData, de sa méthode `applyFilter()`.

Création d'un filtre

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour créer un objet filtre, il suffit d'appeler la fonction constructeur de la classe du filtre voulu. Par exemple, pour créer un objet DropShadowFilter, utilisez le code suivant :

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

Bien que cela n'apparaisse pas dans cet exemple, le constructeur de `DropShadowFilter()` (comme tous les autres constructeurs des classes de filtres) accepte plusieurs paramètres facultatifs qui permettent de modifier l'aspect de l'effet du filtre.

Application d'un filtre

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'objet filtre a été créé, vous pouvez l'appliquer à un objet d'affichage ou à un objet BitmapData, mais le mode d'application du filtre dépend de l'objet concerné.

Application d'un filtre à un objet d'affichage

Pour appliquer un effet de filtrage à un objet d'affichage, utilisez sa propriété `filters`. La propriété `filters` d'un objet d'affichage est une occurrence de l'objet Array, dont les éléments sont les objets filtres appliqués à l'objet d'affichage. Pour appliquer un seul filtre à un objet d'affichage, créez l'occurrence de ce filtre, ajoutez-la à une occurrence d'Array, et affectez cet objet Array à la propriété `filters` de l'objet d'affichage :

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.filters.DropShadowFilter;

// Create a bitmapData object and render it to screen
var myBitmapData:BitmapData = new BitmapData(100,100,false,0xFFFF3300);
var myDisplayObject:Bitmap = new Bitmap(myBitmapData);
addChild(myDisplayObject);

// Create a DropShadowFilter instance.
var dropShadow:DropShadowFilter = new DropShadowFilter();

// Create the filters array, adding the filter to the array by passing it as
// a parameter to the Array() constructor.
var filtersArray:Array = new Array(dropShadow);

// Assign the filters array to the display object to apply the filter.
myDisplayObject.filters = filtersArray;
```

Pour affecter plusieurs filtres à l'objet, il suffit d'ajouter tous ces filtres à l'occurrence d'Array avant de l'affecter à la propriété `filters`. Vous pouvez ajouter plusieurs objets à un objet Array en les passant en paramètres à son constructeur. Par exemple, ce code applique un filtre Biseau et un filtre Rayonnement à l'objet d'affichage créé précédemment :

```
import flash.filters.BevelFilter;
import flash.filters.GlowFilter;

// Create the filters and add them to an array.
var bevel:BevelFilter = new BevelFilter();
var glow:GlowFilter = new GlowFilter();
var filtersArray:Array = new Array(bevel, glow);

// Assign the filters array to the display object to apply the filter.
myDisplayObject.filters = filtersArray;
```

Pour créer le tableau des filtres, vous pouvez utiliser le constructeur `new Array()` (comme dans les exemples précédents) ou la syntaxe littérale Array, en mettant les filtres entre crochets (`[]`). Par exemple, cette ligne de code :

```
var filters:Array = new Array(dropShadow, blur);
```

donne un résultat identique à celle-ci :

```
var filters:Array = [dropShadow, blur];
```

Si vous appliquez plusieurs filtres à des objets d'affichage, ils sont appliqués en séquence, de manière cumulative. Par exemple, si un tableau de filtres comporte deux éléments, le filtre Biseau puis le filtre Ombre portée, ce dernier est appliqué à la fois au filtre Biseau et à l'objet d'affichage lui-même, du fait que le filtre Ombre portée est en seconde position dans le tableau des filtres. Si vous souhaitez appliquer des filtres de manière non cumulative, appliquez chaque filtre à une nouvelle copie de l'objet d'affichage.

Pour affecter uniquement un ou quelques filtres à un objet d'affichage, vous pouvez créer l'occurrence du filtre et l'affecter à l'objet dans la même instruction. Par exemple, le code suivant applique un filtre Flou à un objet d'affichage nommé `myDisplayObject` :

```
myDisplayObject.filters = [new BlurFilter()];
```

Le code précédent crée une occurrence d'Array en utilisant la syntaxe littérale d'Array (entre crochets), puis crée une nouvelle occurrence de BlurFilter comme élément du tableau, et affecte ce dernier à la propriété `filters` de l'objet d'affichage `myDisplayObject`.

Suppression des filtres appliqués à un objet

Pour supprimer tous les filtres d'un objet d'affichage, il suffit d'affecter la valeur null à la propriété `filters` de celui-ci :

```
myDisplayObject.filters = null;
```

Si vous avez appliqué plusieurs filtres à un objet et souhaitez n'en supprimer qu'un, plusieurs étapes sont nécessaires pour modifier le tableau de la propriété `filters`. Pour plus d'informations, voir « Problèmes potentiels d'utilisation des filtres » à la page 279.

Application d'un filtre à un objet BitmapData

L'application d'un filtre à un objet BitmapData nécessite d'utiliser la méthode `applyFilter()` de l'objet BitmapData :

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

La méthode `applyFilter()` applique un filtre à un objet source BitmapData, produisant ainsi une nouvelle image filtrée. Cette méthode ne modifie pas l'image originale, car le résultat de l'application du filtre à celle-ci est enregistré dans l'occurrence de BitmapData pour laquelle la méthode `applyFilter()` est appelée.

Fonctionnement des filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le filtrage des objets d'affichage consiste à mettre en cache une copie de l'objet original sous forme d'un bitmap transparent.

Lorsqu'un filtre a été appliqué à un objet d'affichage, le moteur d'exécution conserve en cache l'objet sous forme de bitmap tant que cet objet possède une liste de filtres valide. Le bitmap source est ensuite repris en tant qu'image originale pour les effets de filtrage suivants.

Tout objet d'affichage comporte généralement deux bitmaps : le premier avec l'objet d'affichage source non filtré d'origine et un autre pour l'image finale après filtrage. L'image finale est utilisée pour le rendu. Tant que l'objet d'affichage ne change pas, l'image source ne nécessite aucune actualisation.

Problèmes potentiels d'utilisation des filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Plusieurs sources potentielles de confusion ou de problèmes peuvent survenir lors de l'utilisation de filtres.

Filtres et mise en cache bitmap

Pour appliquer un filtre à un objet d'affichage, vous devez activer la mise en cache sous forme de bitmap pour cet objet. Si vous appliquez un filtre à un objet d'affichage dont la propriété `cacheAsBitmap` est `false`, la propriété `cacheAsBitmap` de l'objet est automatiquement définie sur `true`. Si vous supprimez tous les filtres appliqués à l'objet d'affichage, la propriété `cacheAsBitmap` retrouve la valeur précédemment définie.

Modification des filtres à l'exécution

Si un ou plusieurs filtres sont déjà appliqués à un objet d'affichage, vous ne pouvez pas modifier le jeu de filtres en ajoutant d'autres filtres ou en supprimant des filtres existants du tableau de la propriété `filters`. Pour modifier le jeu de filtres appliqué ou lui ajouter des filtres, vous devez plutôt effectuer les modifications requises dans un tableau distinct, puis l'assigner à la propriété filters de l'objet d'affichage pour que les filtres soient appliqués à l'objet. La procédure la plus simple consiste à lire le tableau de la propriété `filters` dans une variable Array, puis à effectuer les modifications requises dans ce tableau temporaire. Vous réassignez alors ce tableau à la propriété `filters` de l'objet d'affichage. Dans les cas de figure plus complexes, il peut s'avérer nécessaire de conserver un tableau maître distinct de filtres. Vous effectuez toute modification requise dans ce tableau maître de filtres, puis vous le réassignez à la propriété `filters` de l'objet d'affichage après chaque modification.

Ajout d'un autre filtre

Le code suivant illustre le processus d'ajout d'un autre filtre à un objet d'affichage auquel sont déjà appliqués un ou plusieurs filtres. Initialement, un filtre Rayonnement est appliqué à l'objet d'affichage `myDisplayObject`. Lorsque l'utilisateur clique sur l'objet d'affichage, la fonction `addFilters()` est appelée. Dans cette fonction, deux filtres supplémentaires sont appliqués à `myDisplayObject` :

```
import flash.events.MouseEvent;
import flash.filters.*;

myDisplayObject.filters = [new GlowFilter()];

function addFilters(event:MouseEvent):void
{
    // Make a copy of the filters array.
    var filtersCopy:Array = myDisplayObject.filters;

    // Make desired changes to the filters (in this case, adding filters).
    filtersCopy.push(new BlurFilter());
    filtersCopy.push(new DropShadowFilter());

    // Apply the changes by reassigning the array to the filters property.
    myDisplayObject.filters = filtersCopy;
}

myDisplayObject.addEventListener(MouseEvent.CLICK, addFilters);
```

Suppression d'un filtre dans un jeu de filtres

Si plusieurs filtres sont appliqués à un objet d'affichage et que vous souhaitez supprimer l'un des filtres sans affecter les autres filtres appliqués, vous copiez les filtres dans un tableau temporaire, vous supprimez le filtre superflu du tableau, puis vous réassignez le tableau temporaire à la propriété `filters` de l'objet d'affichage. La section « Récupération des valeurs et suppression des éléments du tableau » à la page 32 décrit plusieurs techniques de suppression d'un ou de plusieurs éléments de tout tableau.

La technique la plus simple consiste à supprimer le filtre en tête de pile appliqué à l'objet (en d'autres termes, le dernier filtre appliqué à ce dernier). Vous utilisez la méthode `pop()` de la classe Array pour supprimer le filtre du tableau :

```
// Example of removing the top-most filter from a display object
// named "filteredObject".

var tempFilters:Array = filteredObject.filters;

// Remove the last element from the Array (the top-most filter).
tempFilters.pop();

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

Pour supprimer le filtre en bas de pile (en d'autres termes, le premier filtre appliqué à l'objet), vous utilisez le même code en substituant la méthode `shift()` de la classe Array à la méthode `pop()`.

Pour supprimer un filtre figurant au centre d'un tableau de filtres (sous réserve que le tableau contienne plus de deux filtres), vous disposez de la méthode `splice()`. Vous devez connaître l'index (la position dans le tableau) du filtre à supprimer. Par exemple, le code suivant supprime le deuxième filtre (doté de l'index 1) d'un objet d'affichage :

```
// Example of removing a filter from the middle of a stack of filters
// applied to a display object named "filteredObject".

var tempFilters:Array = filteredObject.filters;

// Remove the second filter from the array. It's the item at index 1
// because Array indexes start from 0.
// The first "1" indicates the index of the filter to remove; the
// second "1" indicates how many elements to remove.
tempFilters.splice(1, 1);

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

Identification de l'index d'un filtre

Vous devez connaître l'index du filtre à supprimer du tableau. A cet effet, il est nécessaire d'identifier (selon le mode de conception de l'application) ou de calculer l'index du filtre à supprimer.

L'approche recommandée consiste à concevoir votre application de sorte que le filtre à supprimer occupe systématiquement la même position dans le jeu de filtres. Par exemple, si vous disposez d'un objet d'affichage unique auquel sont appliqués un filtre Convolution et un filtre Ombre portée (dans cet ordre) et que vous souhaitez supprimer le filtre Ombre portée tout en conservant le filtre Convolution, vous connaissez la position occupée par le filtre (première). Vous savez donc à l'avance quelle méthode Array utiliser (soit, dans ce cas, `Array.pop()` pour supprimer le filtre Ombre portée).

Si le filtre à supprimer est toujours d'un type déterminé, mais qu'il n'occupe pas systématiquement la même position dans le jeu de filtres, vous pouvez vérifier le type de données de chaque filtre du tableau pour identifier le filtre à supprimer. Par exemple, le code suivant identifie le filtre Rayonnement dans un jeu de filtres et le supprime de ce dernier.

```
// Example of removing a glow filter from a set of filters, where the
//filter you want to remove is the only GlowFilter instance applied
// to the filtered object.

var tempFilters:Array = filteredObject.filters;

// Loop through the filters to find the index of the GlowFilter instance.
var glowIndex:int;
var numFilters:int = tempFilters.length;
for (var i:int = 0; i < numFilters; i++)
{
    if (tempFilters[i] is GlowFilter)
    {
        glowIndex = i;
        break;
    }
}

// Remove the glow filter from the array.
tempFilters.splice(glowIndex, 1);

// Apply the new set of filters to the display object.
filteredObject.filters = tempFilters;
```

Dans un cas de figure plus complexe (si le filtre à supprimer est sélectionné à l'exécution, par exemple), l'approche recommandée consiste à conserver une copie distincte et permanente du tableau de filtres, qui sert de liste maîtresse de filtres. Lorsque vous modifiez le jeu de filtres, modifiez la liste maîtresse, puis appliquez ce tableau de filtres en tant que propriété `filters` de l'objet d'affichage.

Par exemple, dans le code ci-après, plusieurs filtres Convolution sont appliqués à un objet d'affichage pour créer divers effets visuels et l'un de ces filtres est supprimé ultérieurement dans l'application sans affecter les autres filtres. Dans ce cas de figure, le code conserve une copie maîtresse du tableau de filtres, ainsi qu'une référence au filtre à supprimer. Rechercher et identifier le filtre approprié est similaire à l'approche précédente, excepté qu'au lieu d'effectuer une copie temporaire du tableau de filtres, la copie maîtresse est manipulée, puis appliquée à l'objet d'affichage.

```
// Example of removing a filter from a set of
// filters, where there may be more than one
// of that type of filter applied to the filtered
// object, and you only want to remove one.

// A master list of filters is stored in a separate,
// persistent Array variable.
var masterFilterList:Array;

// At some point, you store a reference to the filter you
// want to remove.
var filterToRemove:ConvolutionFilter;

// ... assume the filters have been added to masterFilterList,
// which is then assigned as the filteredObject.filters:
filteredObject.filters = masterFilterList;

// ... later, when it's time to remove the filter, this code gets called:

// Loop through the filters to find the index of masterFilterList.
var removeIndex:int = -1;
var numFilters:int = masterFilterList.length;
for (var i:int = 0; i < numFilters; i++)
{
    if (masterFilterList[i] == filterToRemove)
    {
        removeIndex = i;
        break;
    }
}

if (removeIndex >= 0)
{
    // Remove the filter from the array.
    masterFilterList.splice(removeIndex, 1);

    // Apply the new set of filters to the display object.
    filteredObject.filters = masterFilterList;
}
```

Si vous adoptez cette approche (qui consiste à comparer une référence stockée à un filtre aux éléments que contient le tableau de filtres pour identifier le filtre à supprimer), vous *devez* conserver une copie distincte du tableau de filtres. En effet, le code ne fonctionne pas si vous comparez la référence stockée au filtre aux éléments d'un tableau temporaire copié dans la propriété `filters` de l'objet d'affichage. La raison en est simple : lorsque vous assignez un tableau à la propriété `filters`, le moteur d'exécution effectue une copie de chaque objet filtre du tableau. Ces copies (plutôt que les objets d'origine) sont appliquées à l'objet d'affichage. Lorsque vous lisez la propriété `filters` dans un tableau temporaire, celui-ci contient des références aux objets filtre copiés plutôt qu'aux objets filtre d'origine. Par conséquent, si vous tentez dans l'exemple précédent de déterminer l'index de `filterToRemove` en le comparant aux filtres d'un tableau de filtres temporaire, aucune correspondance n'est détectée.

Filtres et transformations d'objets

Les zones filtrées (ombres portées, par exemple) situées hors du cadre de sélection d'un objet d'affichage ne sont pas prises en considération pour la détection de clics (chevauchement ou intersection de deux occurrences). La méthode de détection des clics de la classe DisplayObject étant de type vectoriel, il est impossible d'en pratiquer une sur le bitmap résultant. Par exemple, si vous appliquez un filtre Biseau à une occurrence de bouton, la détection des clics n'est pas possible sur la partie biseautée de l'occurrence.

Le redimensionnement, la rotation et l'inclinaison ne sont pas pris en charge par les filtres ; si l'objet d'affichage filtré lui-même est redimensionné (`scaleX` et `scaleY` différents de 100 %), l'effet de filtre n'est pas redimensionné avec l'occurrence. La forme originale de l'occurrence est certes pivotée, inclinée ou redimensionnée, mais pas le filtre.

Vous pouvez animer une occurrence avec un filtre afin de créer des effets réalistes, ou imbriquer des occurrences et utiliser la classe BitmapData pour animer des filtres afin d'obtenir ces effets.

Filtres et objets bitmaps

Si vous appliquez un filtre à un objet BitmapData, la propriété `cacheAsBitmap` de cet objet est automatiquement `true`. Le filtre peut ainsi être appliqué à la copie de l'objet plutôt qu'à ce dernier.

Cette copie est alors placée à l'écran par-dessus l'objet original, aussi près que possible, au pixel près. Si les limites du bitmap original changent, la copie à laquelle le filtrage est appliqué est recréée à partir de l'original au lieu d'être étirée.

Si vous supprimez tous les filtres de l'objet d'affichage, la propriété `cacheAsBitmap` retrouve sa valeur d'origine.

Filtres d'affichage disponibles

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 comprend dix classes de filtre que vous pouvez appliquer aux objets d'affichage et aux objets BitmapData :

- filtre Biseau (classe BevelFilter)
- filtre Flou (classe BlurFilter)
- filtre Ombre portée (classe DropShadowFilter)
- filtre Rayonnement (classe GlowFilter)
- filtre Biseau dégradé (classe GradientBevelFilter)
- filtre Rayonnement dégradé (classe GradientGlowFilter)
- filtre Matrice de couleurs (classe ColorMatrixFilter)
- filtre Convolution (classe ConvolutionFilter)
- filtre Mappage de déplacement (classe DisplacementMapFilter)
- filtre Shader (classe ShaderFilter)

Les six premiers sont des filtres simples pouvant être utilisés pour des effets spécifiques, avec certains réglages possibles. Ces six filtres peuvent être appliqués en ActionScript, mais aussi à partir du panneau Filtres de Flash Professional. Par conséquent, même si vous appliquez des filtres à l'aide d'ActionScript, sous réserve de disposer de Flash Professional, vous pouvez utiliser son interface visuelle pour vérifier rapidement l'effet des différents filtres et de leurs réglages afin de créer l'effet désiré.

Les quatre derniers filtres sont uniquement disponibles en ActionScript. Ces filtres (filtre Matrice de couleurs, filtre Convolution, filtre Mappage de déplacement et filtre Shader) permettent de créer des types d'effet beaucoup plus souples. Plutôt que proposer un effet unique optimisé, ils assurent puissance et souplesse. Par exemple, en choisissant différentes valeurs pour sa matrice, il est possible d'utiliser le filtre Convolution pour créer des effets de flou, de gravure, d'accentuation, mais aussi pour des transformations, la détection de contour des couleurs, etc.

Chacun de ces filtres, qu'il soit simple ou complexe, dispose de propriétés dont les valeurs peuvent être modifiées. En général, il existe deux possibilités pour définir les propriétés d'un filtre. Tous les filtres permettent de définir leurs propriétés en passant des valeurs en paramètres au constructeur de l'objet filtre. Que les propriétés du filtre soient définies ou non par un passage de paramètres, il est aussi possible de modifier ultérieurement ses réglages en changeant les valeurs des propriétés de l'objet filtre. La plupart des exemples de code définissent directement les propriétés pour faciliter la compréhension de l'exemple. Néanmoins, il est également possible, en général, d'obtenir le même résultat avec moins de lignes de code, en passant les valeurs en paramètres dans le constructeur de l'objet filtre. Pour plus d'informations sur les caractéristiques de chaque filtre, ses propriétés et les paramètres constructeurs correspondants, voir les codes associés au package flash.filters dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Filtre Biseau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe BevelFilter vous permet d'ajouter une bordure en relief à l'objet filtré. Avec ce filtre, les angles et les côtés des objets semblent ciselés, biseautés.

Les propriétés de la classe BevelFilter permettent de modifier l'apparence du biseau. Vous pouvez définir les couleurs des zones claires et sombres, l'adoucissement et les angles des côtés du biseau, ainsi que la tille de ces derniers. Vous pouvez même créer un effet de poinçonnage.

L'exemple suivant charge une image externe et lui applique un filtre Biseau.

```
import flash.display.*;
import flash.filters.BevelFilter;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.net.URLRequest;

// Load an image onto the Stage.
var imageLoader:Loader = new Loader();
var url:String = "http://www.helpexamples.com/flash/images/image3.jpg";
var urlReq:URLRequest = new URLRequest(url);
imageLoader.load(urlReq);
addChild(imageLoader);

// Create the bevel filter and set filter properties.
var bevel:BevelFilter = new BevelFilter();

bevel.distance = 5;
bevel.angle = 45;
bevel.highlightColor = 0xFFFF00;
bevel.highlightAlpha = 0.8;
bevel.shadowColor = 0x666666;
bevel.shadowAlpha = 0.8;
bevel.blurX = 5;
bevel.blurY = 5;
bevel.strength = 5;
bevel.quality = BitmapFilterQuality.HIGH;
bevel.type = BitmapFilterType.INNER;
bevel.knockout = false;

// Apply filter to the image.
imageLoader.filters = [bevel];
```

Filtre Flou

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe BlurFilter ajoute un effet de flou à un objet d'affichage et son contenu. Les effets de flou permettent de donner l'impression qu'un objet n'est pas dans le plan de mise au point ou de simuler l'effet d'un mouvement rapide (flou de mouvement). En choisissant une valeur faible pour la propriété `quality`, vous pouvez simuler un effet de photo légèrement floue. Le choix d'une valeur élevée pour la propriété `quality` permet d'obtenir un effet de flou léger proche de celui d'un flou gaussien.

L'exemple suivant crée un objet cercle à l'aide de la méthode `drawCircle()` de la classe Graphics, puis lui applique un filtre Flou :

```
import flash.display.Sprite;
import flash.filters.BitmapFilterQuality;
import flash.filters.BlurFilter;

// Draw a circle.
var redDotCutout:Sprite = new Sprite();
redDotCutout.graphics.lineStyle();
redDotCutout.graphics.beginFill(0xFF0000);
redDotCutout.graphics.drawCircle(145, 90, 25);
redDotCutout.graphics.endFill();

// Add the circle to the display list.
addChild(redDotCutout);

// Apply the blur filter to the rectangle.
var blur:BlurFilter = new BlurFilter();
blur.blurX = 10;
blur.blurY = 10;
blur.quality = BitmapFilterQuality.MEDIUM;
redDotCutout.filters = [blur];
```

Filtre Ombre portée

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'ombre portée donne l'impression d'une source lumineuse ponctuelle au-dessus de l'objet cible. Il est possible de modifier la position et l'intensité de cette source lumineuse pour produire divers effets d'ombre portée.

La classe DropShadowFilter utilise un algorithme similaire à celui du filtre Flou. La principale différence tient au fait que le filtre Ombre portée possède quelques propriétés supplémentaires qui permettent de simuler diverses caractéristiques d'une source lumineuse (canal alpha, couleur, décalage et luminosité).

Le filtre Ombre portée permet aussi d'appliquer des options de transformation au style de l'ombre portée (ombre interne ou externe, ou mode de masquage).

Le code suivant crée un objet Sprite carré et lui applique un filtre Ombre portée.

```
import flash.display.Sprite;
import flash.filters.DropShadowFilter;

// Draw a box.
var boxShadow:Sprite = new Sprite();
boxShadow.graphics.lineStyle(1);
boxShadow.graphics.beginFill(0xFF3300);
boxShadow.graphics.drawRect(0, 0, 100, 100);
boxShadow.graphics.endFill();
addChild(boxShadow);

// Apply the drop shadow filter to the box.
var shadow:DropShadowFilter = new DropShadowFilter();
shadow.distance = 10;
shadow.angle = 25;

// You can also set other properties, such as the shadow color,
// alpha, amount of blur, strength, quality, and options for
// inner shadows and knockout effects.

boxShadow.filters = [shadow];
```

Filtre Rayonnement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe GlowFilter applique un effet d'éclairage aux objets d'affichage afin de suggérer qu'une lumière est braquée à partir du dessous de l'objet pour créer un faible rayonnement.

Comme le filtre Ombre portée, le filtre Rayonnement possède des propriétés qui permettent de modifier la distance, l'angle et la couleur de la source lumineuse en fonction de l'effet désiré. L'objet GlowFilter comporte aussi plusieurs options pour modifier le style de rayonnement, notamment le rayonnement interne ou externe et le mode de masquage.

Le code suivant crée un objet Sprite en forme de croix et lui applique un filtre Rayonnement.

```
import flash.display.Sprite;
import flash.filters.BitmapFilterQuality;
import flash.filters.GlowFilter;

// Create a cross graphic.
var crossGraphic:Sprite = new Sprite();
crossGraphic.graphics.lineStyle();
crossGraphic.graphics.beginFill(0xCCCC00);
crossGraphic.graphics.drawRect(60, 90, 100, 20);
crossGraphic.graphics.drawRect(100, 50, 20, 100);
crossGraphic.graphics.endFill();
addChild(crossGraphic);

// Apply the glow filter to the cross shape.
var glow:GlowFilter = new GlowFilter();
glow.color = 0x009922;
glow.alpha = 1;
glow.blurX = 25;
glow.blurY = 25;
glow.quality = BitmapFilterQuality.MEDIUM;

crossGraphic.filters = [glow];
```

Filtre Biseau dégradé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe GradientBevelFilter vous permet d'appliquer un effet de biseau optimisé aux objets d'affichage ou aux objets BitmapData. L'utilisation d'un dégradé de couleurs sur le biseau améliore beaucoup l'effet de relief de celui-ci, en donnant aux côtés un aspect 3D plus réaliste.

L'exemple suivant crée un objet rectangle à l'aide de la méthode `drawRect()` de la classe Shape, puis lui applique un filtre Biseau dégradé :

```
import flash.display.Shape;
import flash.filters.BitmapFilterQuality;
import flash.filters.GradientBevelFilter;

// Draw a rectangle.
var box:Shape = new Shape();
box.graphics.lineStyle();
box.graphics.beginFill(0xFEFE78);
box.graphics.drawRect(100, 50, 90, 200);
box.graphics.endFill();

// Apply a gradient bevel to the rectangle.
var gradientBevel:GradientBevelFilter = new GradientBevelFilter();

gradientBevel.distance = 8;
gradientBevel.angle = 225; // opposite of 45 degrees
gradientBevel.colors = [0xFFFFCC, 0xFEFE78, 0x8F8E01];
gradientBevel.alphas = [1, 0, 1];
gradientBevel.ratios = [0, 128, 255];
gradientBevel.blurX = 8;
gradientBevel.blurY = 8;
gradientBevel.quality = BitmapFilterQuality.HIGH;

// Other properties let you set the filter strength and set options
// for inner bevel and knockout effects.

box.filters = [gradientBevel];

// Add the graphic to the display list.
addChild(box);
```

Filtre Rayonnement dégradé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe GradientGlowFilter vous permet d'appliquer un effet de rayonnement optimisé aux objets d'affichage ou aux objets BitmapData. Cet effet donne davantage de contrôle sur le rayonnement, produisant ainsi un effet plus réaliste. De plus, le filtre Rayonnement dégradé permet d'appliquer un rayonnement aux côtés intérieur, extérieur ou supérieur de l'objet.

L'exemple suivant dessine un cercle auquel un filtre Rayonnement dégradé est appliqué. Le montant de flou horizontal et vertical augmente à mesure que le pointeur de la souris s'approche du coin inférieur droit de la scène. Si l'utilisateur clique dans la scène, le niveau de flou augmente.

```
import flash.events.MouseEvent;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.filters.GradientGlowFilter;

// Create a new Shape instance.
var shape:Shape = new Shape();

// Draw the shape.
shape.graphics.beginFill(0xFF0000, 100);
shape.graphics.moveTo(0, 0);
shape.graphics.lineTo(100, 0);
shape.graphics.lineTo(100, 100);
shape.graphics.lineTo(0, 100);
shape.graphics.lineTo(0, 0);
shape.graphics.endFill();

// Position the shape on the Stage.
addChild(shape);
shape.x = 100;
shape.y = 100;

// Define a gradient glow.
var gradientGlow:GradientGlowFilter = new GradientGlowFilter();
gradientGlow.distance = 0;
gradientGlow.angle = 45;
gradientGlow.colors = [0x000000, 0xFF0000];
gradientGlow.alphas = [0, 1];
gradientGlow.ratios = [0, 255];
gradientGlow.blurX = 10;
gradientGlow.blurY = 10;
gradientGlow.strength = 2;
gradientGlow.quality = BitmapFilterQuality.HIGH;
gradientGlow.type = BitmapFilterType.OUTER;

// Define functions to listen for two events.
function onClick(event:MouseEvent):void
{
    gradientGlow.strength++;
    shape.filters = [gradientGlow];
}

function onMouseMove(event:MouseEvent):void
{
    gradientGlow.blurX = (stage.mouseX / stage.stageWidth) * 255;
    gradientGlow.blurY = (stage.mouseY / stage.stageHeight) * 255;
    shape.filters = [gradientGlow];
}
stage.addEventListener(MouseEvent.CLICK, onClick);
stage.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);
```

Exemple : Combinaison de filtres de base

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple de code suivant applique plusieurs filtres de base, en combinaison avec un timer pour la création d'actions répétitives, pour obtenir la simulation d'un feu de circulation.

```
import flash.display.Shape;
import flash.events.TimerEvent;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
import flash.filters.GradientBevelFilter;
import flash.utils.Timer;

var count:Number = 1;
var distance:Number = 8;
var angleInDegrees:Number = 225; // opposite of 45 degrees
var colors:Array = [0xFFFFCC, 0xFEFE78, 0x8F8E01];
var alphas:Array = [1, 0, 1];
var ratios:Array = [0, 128, 255];
var blurX:Number = 8;
var blurY:Number = 8;
var strength:Number = 1;
var quality:Number = BitmapFilterQuality.HIGH;
var type:String = BitmapFilterType.INNER;
var knockout:Boolean = false;

// Draw the rectangle background for the traffic light.
var box:Shape = new Shape();
box.graphics.lineStyle();
box.graphics.beginFill(0xFEFE78);
box.graphics.drawRect(100, 50, 90, 200);
box.graphics.endFill();

// Draw the 3 circles for the three lights.
var stopLight:Shape = new Shape();
stopLight.graphics.lineStyle();
stopLight.graphics.beginFill(0xFF0000);
stopLight.graphics.drawCircle(145,90,25);
stopLight.graphics.endFill();

var cautionLight:Shape = new Shape();
cautionLight.graphics.lineStyle();
cautionLight.graphics.beginFill(0xFF9900);
cautionLight.graphics.drawCircle(145,150,25);
cautionLight.graphics.endFill();

var goLight:Shape = new Shape();
goLight.graphics.lineStyle();
goLight.graphics.beginFill(0x00CC00);
goLight.graphics.drawCircle(145,210,25);
goLight.graphics.endFill();

// Add the graphics to the display list.
addChild(box);
```

```
addChild(stopLight);
addChild(cautionLight);
addChild(goLight);

// Apply a gradient bevel to the traffic light rectangle.
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(distance, angleInDegrees,
colors, alphas, ratios, blurX, blurY, strength, quality, type, knockout);
box.filters = [gradientBevel];

// Create the inner shadow (for lights when off) and glow
// (for lights when on).
var innerShadow:DropShadowFilter = new DropShadowFilter(5, 45, 0, 0.5, 3, 3, 1, 1, true,
false);
var redGlow:GlowFilter = new GlowFilter(0xFF0000, 1, 30, 30, 1, 1, false, false);
var yellowGlow:GlowFilter = new GlowFilter(0xFF9900, 1, 30, 30, 1, 1, false, false);
var greenGlow:GlowFilter = new GlowFilter(0x00CC00, 1, 30, 30, 1, 1, false, false);

// Set the starting state of the lights (green on, red/yellow off).
stopLight.filters = [innerShadow];
cautionLight.filters = [innerShadow];
goLight.filters = [greenGlow];

// Swap the filters based on the count value.
function trafficControl(event:TimerEvent):void
{
    if (count == 4)
    {
        count = 1;
    }

    switch (count)
    {
        case 1:
            stopLight.filters = [innerShadow];
            cautionLight.filters = [yellowGlow];
            goLight.filters = [innerShadow];
            break;
        case 2:
            stopLight.filters = [redGlow];
            cautionLight.filters = [innerShadow];
            goLight.filters = [innerShadow];
            break;
        case 3:
            stopLight.filters = [innerShadow];
            cautionLight.filters = [innerShadow];
            goLight.filters = [greenGlow];
            break;
    }

    count++;
}

// Create a timer to swap the filters at a 3 second interval.
var timer:Timer = new Timer(3000, 9);
timer.addEventListener(TimerEvent.TIMER, trafficControl);
timer.start();
```

Filtre Matrice de couleurs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ColorMatrixFilter permet de manipuler les valeurs de couleur et les valeurs alpha des objets filtrés. Il est ainsi possible de créer des changements de saturation, des rotations de teinte (passage d'une palette d'une plage de couleur à une autre), de définir la luminance de la couche alpha et de produire d'autres effets de manipulation des couleurs en utilisant les valeurs d'un canal couleur pour les appliquer aux autres canaux.

Le principe de fonctionnement de ce filtre est le suivant : les pixels de l'image source sont analysés un par un et leurs composants rouge, vert, bleu et alpha sont séparés. Les valeurs de la matrice de couleur sont alors multipliées par chacune de ces valeurs, et les résultats sont ajoutés pour déterminer la valeur colorimétrique résultante qui sera affichée à l'écran pour ce pixel. La propriété `matrix` du filtre est un tableau de 20 nombres qui sont utilisés pour le calcul de la couleur finale. Pour plus d'informations sur l'algorithme utilisé pour calculer les valeurs de couleur, voir la description de la propriété `matrix` de la classe ColorMatrixFilter dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Filtre Convolution

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ConvolutionFilter permet d'appliquer un large éventail de transformations de traitement d'images aux objets BitmapData ou aux objets d'affichage, tels que la définition du flou, la détection du contour, l'accentuation, l'estampage et le biseautage.

Le principe de fonctionnement du filtre Convolution est le suivant : les pixels de l'image source sont analysés un par un pour déterminer la couleur finale de ce pixel sur la base de sa valeur et de celles des pixels adjacents. Une matrice, sous forme d'un tableau de valeurs numériques, indique dans quelle mesure la valeur de chaque pixel adjacent particulier affecte la valeur finale.

Le type de matrice le plus fréquemment utilisé est un tableau de trois par trois. La matrice comporte donc neuf valeurs :

```
N   N   N
N   P   N
N   N   N
```

Lorsque le filtre Convolution est appliqué à un pixel donné, il analyse la valeur colorimétrique du pixel lui-même (« P » dans notre exemple) et celles des pixels environnants (« N » dans l'exemple). Toutefois, le choix des valeurs de la matrice permet de spécifier la priorité de certains pixels pour le calcul de l'image résultante.

Par exemple, la matrice suivante, appliquée à un filtre Convolution, laissera l'image intacte, exactement comme l'image originale :

```
0   0   0
0   1   0
0   0   0
```

L'image n'a pas été modifiée car la valeur du pixel original a une priorité relative de 1 pour déterminer la couleur du pixel final, alors que les pixels environnants ont une priorité relative de 0 (autrement dit, leur couleur n'affecte pas l'image finale).

De même, la matrice suivante provoque un décalage à gauche de tous les pixels d'une image :

```
0   0   0
0   0   1
0   0   0
```

Notez que dans cet exemple, le pixel lui-même n'a aucun effet sur la valeur finale du pixel affiché au même emplacement dans l'image finale : seule la valeur du pixel de droite est utilisée pour déterminer la valeur résultante de chaque pixel.

En ActionScript, la matrice est une combinaison d'une occurrence de l'objet Array contenant les valeurs et deux propriétés spécifiant le nombre de lignes et de colonnes de la matrice. L'exemple suivant charge une image, puis lui applique un filtre Convolution sur la base de la matrice du listing précédent :

```
// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

function applyFilter(event:MouseEvent):void
{
    // Create the convolution matrix.
    var matrix:Array = [0, 0, 0,
                                    0, 0, 1,
                                    0, 0, 0];

    var convolution:ConvolutionFilter = new ConvolutionFilter();
    convolution.matrixX = 3;
    convolution.matrixY = 3;
    convolution.matrix = matrix;
    convolution.divisor = 1;

    loader.filters = [convolution];
}

loader.addEventListener(MouseEvent.CLICK, applyFilter);
```

Un point important n'est pas évident dans ce code : l'effet de valeurs autres que 1 ou 0 dans la matrice. Par exemple, la même matrice avec le chiffre 8 au lieu de 1 dans le coin supérieur droit effectuerait la même action (décalage des pixels vers la gauche). Toutefois, les couleurs de l'image seraient 8 fois plus lumineuses. En effet, les valeurs finales de couleur des pixels sont calculées en multipliant les valeurs de la matrice par celles des couleurs originales des pixels, en additionnant ces valeurs, puis en les divisant par celle de la propriété `divisor` du filtre. Notez que, dans cet exemple, la propriété `divisor` a la valeur 1. En règle générale, pour que la luminosité des couleurs reste à peu près identique à celle des couleurs de l'image originale, la propriété divisor doit avoir une valeur égale à la somme des valeurs de la matrice. Ainsi, avec une matrice dont la somme des valeurs est 8, et pour un diviseur de 1, l'image résultante sera environ 8 fois plus lumineuse que l'image originale.

Bien que l'effet de cette matrice ne soit pas très spectaculaire, d'autres valeurs peuvent être utilisées pour créer divers effets. Voici quelques ensembles standard de valeurs de matrice permettant d'obtenir divers effets avec une matrice de trois sur trois :

- Flou de base (diviseur 5) :

```
0 1 0
1 1 1
0 1 0
```

- Accentuation (diviseur 1) :

```
 0, -1, 0
-1, 5, -1
 0, -1, 0
```

- Détection des contours (diviseur 1) :

```
 0, -1, 0
-1, 4, -1
 0, -1, 0
```

- Estampage (diviseur 1) :

```
-2, -1, 0
-1, 1, 1
 0, 1, 2
```

  Notez qu'avec la plupart de ces effets, le diviseur est 1. En effet, l'addition des valeurs négatives et des valeurs positives dans la matrice donne 1 (ou 0 dans le cas de la détection des contours, mais la valeur de la propriété `divisor` ne peut pas être 0).

Filtre Mappage de déplacement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe DisplacementMapFilter utilise des valeurs de pixel extraites d'un objet BitmapData (appelé image de mappage du déplacement) pour appliquer un effet de déplacement à un nouvel objet. L'image de mappage du déplacement est en général différente de l'occurrence d'objet d'affichage ou BitmapData à laquelle le filtre est appliqué. L'effet de déplacement nécessite de déplacer les pixels de l'image filtrée, autrement dit de les décaler d'un certain niveau. Ce filtre permet de créer un effet de décalage, de gondole ou de marbrure.

La direction et la valeur du déplacement appliqué à un pixel donné sont déterminées par la valeur colorimétrique de l'image de mappage du déplacement. Pour utiliser ce filtre, il est nécessaire de spécifier l'image de mappage, ainsi que les valeurs suivantes, qui permettent de contrôler le calcul du déplacement :

- Point de mappage : emplacement, dans l'image filtrée, auquel le coin supérieur gauche du filtre de déplacement sera appliqué. Ce paramètre n'est nécessaire que pour appliquer le filtre à une partie de l'image seulement.
- Composant X : canal couleur de l'image de mappage qui affecte la position x des pixels.
- Composant Y : canal couleur de l'image de mappage qui affecte la position y des pixels.
- Echelle X : valeur multiplicatrice qui indique le niveau de déplacement sur l'axe x.
- Echelle Y : valeur multiplicatrice qui indique le niveau de déplacement sur l'axe y.
- Mode de filtrage : détermine la marche à suivre dans le cas d'espaces vides créés par le décalage des pixels. Les options, définies par des constantes dans la classe DisplacementMapFilterMode, sont d'afficher les pixels originaux (mode `IGNORE`), de décaler et transférer les pixels de l'autre côté de l'image (mode `WRAP`, qui est le mode par défaut), d'utiliser le pixel déplacé le plus proche (mode `CLAMP`) ou de remplir ces espaces vides avec une couleur (mode `COLOR`).

Pour comprendre le fonctionnement d'un filtre de mappage de déplacement, prenons un exemple simple. Dans le code ci-dessous, une image est chargée, puis elle est centrée sur la scène et un filtre Mappage de déplacement lui est appliqué, ce qui déplace horizontalement (vers la gauche) les pixels de toute l'image.

```
import flash.display.BitmapData;
import flash.display.Loader;
import flash.events.MouseEvent;
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.net.URLRequest;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image3.jpg");
loader.load(url);
this.addChild(loader);

var mapImage:BitmapData;
var displacementMap:DisplacementMapFilter;

// This function is called when the image finishes loading.
function setupStage(event:Event):void
{
    // Center the loaded image on the Stage.
    loader.x = (stage.stageWidth - loader.width) / 2;
    loader.y = (stage.stageHeight - loader.height) / 2;

    // Create the displacement map image.
    mapImage = new BitmapData(loader.width, loader.height, false, 0xFF0000);

    // Create the displacement filter.
    displacementMap = new DisplacementMapFilter();
    displacementMap.mapBitmap = mapImage;
    displacementMap.mapPoint = new Point(0, 0);
    displacementMap.componentX = BitmapDataChannel.RED;
    displacementMap.scaleX = 250;
    loader.filters = [displacementMap];
}

loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setupStage);
```

Les propriétés utilisées pour définir le déplacement sont les suivantes :

- Bitmap de mappage : le bitmap de déplacement est une nouvelle occurrence de BitmapData, créée par code. Ses dimensions sont identiques à celles de l'image chargée (le déplacement est donc appliqué à toute l'image). Elle est remplie de pixels rouges opaques.
- Point de mappage : cette valeur est définie pour le point 0, 0 (ici encore, le déplacement sera appliqué à toute l'image).
- Composant X : cette valeur reçoit la constante `BitmapDataChannel.RED`, ce qui signifie que c'est la valeur de rouge de l'image bitmap de mappage qui déterminera le niveau de déplacement des pixels sur l'axe x.
- Echelle X : cette valeur est réglée sur 250. L'image de mappage étant entièrement rouge, la valeur totale de déplacement ne décale l'image que faiblement (environ un demi-pixel). Si cette valeur était de 1, l'image ne serait donc décalée que de 0,5 pixel horizontalement. En choisissant une valeur de 250, nous déplaçons l'image d'environ 125 pixels.

Ces valeurs provoquent un déplacement des pixels de l'image filtrée de 250 pixels à gauche. La direction (gauche ou droite) et l'importance du déplacement dépendent de la valeur colorimétrique des pixels de l'image de mappage. Le principe de fonctionnement de ce filtre est le suivant : il analyse un par un les pixels de l'image filtrée (ou tout au moins ceux de la zone à laquelle le filtre est appliqué, ce qui ici signifie toute l'image), et procède comme suit pour chaque pixel :

1. Il détermine le pixel correspondant dans l'image de mappage. Par exemple, pour calculer la valeur de déplacement du pixel du coin supérieur gauche de l'image filtrée, le filtre analyse le pixel correspondant dans le coin supérieur gauche de l'image de mappage.
2. Il détermine la valeur du canal de couleur spécifié dans le pixel de mappage. Dans cet exemple, le canal de couleur du composant x est le rouge. Le filtre recherche donc la valeur du canal rouge au point en question dans l'image de mappage. L'image de mappage étant un rouge opaque, le canal rouge du pixel a la valeur 0xFF, soit 255. Cette valeur est la valeur de déplacement.
3. Ils comparent ensuite la valeur de déplacement à la valeur médiane (127, à mi-chemin entre 0 et 255). Si la valeur de déplacement est inférieure à la valeur médiane, le pixel est déplacé dans une direction positive (vers la droite pour l'axe x, vers le bas pour l'axe y). Par contre, si la valeur de déplacement est supérieure à la valeur médiane (comme dans cet exemple), le pixel est déplacé dans une direction négative (vers la gauche pour l'axe x, vers le haut pour l'axe y). Plus précisément, le filtre soustrait la valeur de déplacement de 127, et le résultat (positif ou négatif) est la valeur relative de déplacement qui est appliquée.
4. Enfin, ils déterminent la valeur réelle de déplacement en calculant le pourcentage de déplacement complet représenté par la valeur de déplacement relatif. Dans cet exemple, un rouge à 100 % provoque un déplacement de 100 %. Ce pourcentage est ensuite multiplié par la valeur de l'échelle x ou de l'échelle y pour déterminer le nombre de pixels de déplacement à appliquer. Dans cet exemple, la valeur de déplacement est 100 % multiplié par un multiple de 250, soit environ 125 pixels à gauche.

Comme nous ne spécifions aucune valeur pour les composants x et y, les valeurs par défaut (qui ne provoquent pas de déplacement) sont utilisées. C'est pourquoi l'image n'est pas déplacée dans le sens vertical.

Dans cet exemple, le paramètre par défaut de mode de filtrage, `WRAP`, est utilisé, si bien que lorsque les pixels sont déplacés vers la gauche, l'espace laissé vide à droite est rempli par les pixels qui ont été décalés le long du côté gauche de l'image. Vous pouvez modifier cette valeur pour voir les différents effets ainsi obtenus. Par exemple, si vous ajoutez la ligne suivante dans la partie du code qui définit les propriétés de déplacement (avant la ligne `loader.filters = [displacementMap]`), l'image semble avoir subi un balayage :

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

Le code ci-dessous propose un exemple plus complexe, en utilisant un filtre Mappage de déplacement pour créer un effet de loupe dans l'image :

```
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.display.BitmapDataChannel;
import flash.display.GradientType;
import flash.display.Loader;
import flash.display.Shape;
import flash.events.MouseEvent;
import flash.filters.DisplacementMapFilter;
import flash.filters.DisplacementMapFilterMode;
import flash.geom.Matrix;
import flash.geom.Point;
import flash.net.URLRequest;

// Create the gradient circles that will together form the
// displacement map image
var radius:uint = 50;

var type:String = GradientType.LINEAR;
var redColors:Array = [0xFF0000, 0x000000];
var blueColors:Array = [0x0000FF, 0x000000];
var alphas:Array = [1, 1];
var ratios:Array = [0, 255];
var xMatrix:Matrix = new Matrix();
xMatrix.createGradientBox(radius * 2, radius * 2);
var yMatrix:Matrix = new Matrix();
yMatrix.createGradientBox(radius * 2, radius * 2, Math.PI / 2);

var xCircle:Shape = new Shape();
xCircle.graphics.lineStyle(0, 0, 0);
xCircle.graphics.beginGradientFill(type, redColors, alphas, ratios, xMatrix);
xCircle.graphics.drawCircle(radius, radius, radius);

var yCircle:Shape = new Shape();
yCircle.graphics.lineStyle(0, 0, 0);
yCircle.graphics.beginGradientFill(type, blueColors, alphas, ratios, yMatrix);
yCircle.graphics.drawCircle(radius, radius, radius);

// Position the circles at the bottom of the screen, for reference.
this.addChild(xCircle);
xCircle.y = stage.stageHeight - xCircle.height;
this.addChild(yCircle);
yCircle.y = stage.stageHeight - yCircle.height;
yCircle.x = 200;

// Load an image onto the Stage.
var loader:Loader = new Loader();
var url:URLRequest = new URLRequest("http://www.helpexamples.com/flash/images/image1.jpg");
loader.load(url);
this.addChild(loader);

// Create the map image by combining the two gradient circles.
var map:BitmapData = new BitmapData(xCircle.width, xCircle.height, false, 0x7F7F7F);
map.draw(xCircle);
var yMap:BitmapData = new BitmapData(yCircle.width, yCircle.height, false, 0x7F7F7F);
yMap.draw(yCircle);
map.copyChannel(yMap, yMap.rect, new Point(0, 0), BitmapDataChannel.BLUE,
BitmapDataChannel.BLUE);
```

```
yMap.dispose();

// Display the map image on the Stage, for reference.
var mapBitmap:Bitmap = new Bitmap(map);
this.addChild(mapBitmap);
mapBitmap.x = 400;
mapBitmap.y = stage.stageHeight - mapBitmap.height;

// This function creates the displacement map filter at the mouse location.
function magnify():void
{
    // Position the filter.
    var filterX:Number = (loader.mouseX) - (map.width / 2);
    var filterY:Number = (loader.mouseY) - (map.height / 2);
    var pt:Point = new Point(filterX, filterY);
    var xyFilter:DisplacementMapFilter = new DisplacementMapFilter();
    xyFilter.mapBitmap = map;
    xyFilter.mapPoint = pt;
    // The red in the map image will control x displacement.
    xyFilter.componentX = BitmapDataChannel.RED;
    // The blue in the map image will control y displacement.
    xyFilter.componentY = BitmapDataChannel.BLUE;
    xyFilter.scaleX = 35;
    xyFilter.scaleY = 35;
    xyFilter.mode = DisplacementMapFilterMode.IGNORE;
    loader.filters = [xyFilter];
}

// This function is called when the mouse moves. If the mouse is
// over the loaded image, it applies the filter.
function moveMagnifier(event:MouseEvent):void
{
    if (loader.hitTestPoint(loader.mouseX, loader.mouseY))
    {
        magnify();
    }
}
loader.addEventListener(MouseEvent.MOUSE_MOVE, moveMagnifier);
```

Ce code génère d'abord deux cercles en dégradé, qui sont combinés pour former l'image de mappage du déplacement. Le cercle rouge est à l'origine du déplacement sur l'axe x (`xyFilter.componentX = BitmapDataChannel.RED`) et le cercle bleu est à l'origine du déplacement sur l'axe y (`xyFilter.componentY = BitmapDataChannel.BLUE`). Pour vous permettre de comprendre plus aisément l'aspect de l'image de mappage du déplacement, le code affiche les cercles originaux, ainsi que le cercle combiné qui fait office d'image de mappage, en bas de l'écran.

Le code charge ensuite une image et, en fonction des déplacements de la souris, applique le filtre de déplacement à la partie de l'image qui est sous la souris. Les cercles dégradés utilisés pour l'image de mappage de déplacement provoquent un effet centrifuge dans la zone à laquelle le filtre est appliqué. Notez que les zones en gris de l'image de mappage du déplacement ne provoquent pas de déplacement. En effet, la valeur du gris est `0x7F7F7F`. Les canaux bleu et rouge de ce niveau de gris ont donc exactement une valeur médiane, si bien qu'il n'y a pas de déplacement lorsqu'une zone grise apparaît dans l'image de mappage. Il n'y a pas non plus de déplacement au centre du cercle. Bien que cette zone ne soit pas de couleur grise, ses canaux rouge et bleu ont des valeurs identiques à celles des canaux rouge et bleu du gris moyen, et puisque le déplacement est basé sur les valeurs de bleu et de rouge, aucun déplacement n'a lieu.

Filtre Shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La classe ShaderFilter vous permet d'utiliser un effet de filtre personnalisé défini en tant que shader de Pixel Bender. Parce que l'effet de filtre est écrit en tant que shader de Pixel Bender, il peut être entièrement personnalisé. Le contenu filtré est transmis au shader en tant qu'image d'entrée et le résultat de l'opération du shader devient le résultat du filtre.

Remarque : *le filtre Shader est pris en charge dans ActionScript à partir de Flash Player 10 et Adobe AIR 1.5.*

Pour appliquer un filtre Shader à un objet, vous devez commencer par créer une occurrence de Shader représentant le shader de Pixel Bender en cours d'utilisation. Pour plus d'informations concernant la procédure de création d'une occurrence de Shader et la définition d'une image d'entrée et des paramètres correspondants, voir « Utilisation des shaders de Pixel Bender » à la page 310.

Lorsque vous utilisez un shader en tant que filtre, vous devez tenir compte de trois considérations importantes :

- Le shader doit être défini pour accepter au moins une image d'entrée.

- L'objet filtré (objet d'affichage ou objet BitmapData auquel est appliqué le filtre) est transmis au shader en tant que première valeur d'image d'entrée. De ce fait, ne définissez pas manuellement la valeur de la première image d'entrée.
- Si le shader définit plusieurs images d'entrée, les autres entrées doivent être stipulées manuellement (en définissant la propriété `input` de toute occurrence de ShaderInput appartenant à l'occurrence de Shader).

Lorsque vous disposez d'un objet Shader pour le shader, vous créez une occurrence de ShaderFilter. Il s'agit de l'objet filtre en tant que tel utilisé comme tout autre filtre. Pour créer un élément ShaderFilter qui utilise un objet Shader, appelez le constructeur `ShaderFilter()` et transmettez l'objet Shader en tant qu'argument, comme illustré dans le code suivant :

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

Pour disposer d'un exemple complet d'utilisation d'un filtre Shader, voir « Utilisation d'un shader comme filtre » à la page 328.

Exemple de filtrage des objets d'affichage : Filter Workbench

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple Filter Workbench comporte une interface utilisateur qui permet d'appliquer divers filtres à des images et autre contenu visuel et de voir le code résultant, qui peut être utilisé pour générer le même effet en ActionScript. Non seulement cette application fournit un outil permettant d'expérimenter avec les filtres, mais elle illustre également les techniques suivantes :

- Création d'occurrences de filtres divers
- Application de plusieurs filtres à un objet d'affichage

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application Filter Workbench résident dans le dossier Samples/FilterWorkbench. L'application se compose des fichiers suivants :

Fichier	Description
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	Classe qui fournit la principale fonctionnalité de l'application, notamment permuter le contenu auquel sont appliqués les filtres et appliquer les filtres au contenu.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	Interface qui définit les méthodes courantes et implémentées par chacune des classes usine de filtres. Cette interface définit la fonctionnalité commune utilisée par la classe FilterWorkbenchController pour interagir avec chaque classe usine de filtres.
Dans le dossier com/example/programmingas3/filterWorkbench/ : BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	Jeu de classes, dont chacune implémente l'interface IFilterFactory. Chacune de ces classes permet de créer et de définir les valeurs associées à un type unique de filtre. Les panneaux de propriétés des filtres de l'application utilisent ces classes usine pour créer des occurrences des filtres correspondants, que la classe FilterWorkbenchController extrait et applique au contenu d'image.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	Interface qui définit les méthodes courantes et implémentées par les classes qui spécifient les panneaux de l'interface utilisateur employés pour manipuler les valeurs de filtre dans l'application.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	Classe d'utilitaires qui comporte une méthode de conversion d'une valeur de couleur numérique au format chaîne hexadécimal.
com/example/programmingas3/filterWorkbench/GradientColor.as	Classe servant d'objet de valeur, qui combine en un objet unique les trois valeurs (couleur, alpha et rapport) associées à chaque couleur dans GradientBevelFilter et GradientGlowFilter.
Interface utilisateur (Flex)	
FilterWorkbench.mxml	Fichier principal qui définit l'interface utilisateur de l'application.
flexapp/FilterWorkbench.as	Classe qui assure la fonctionnalité de l'interface utilisateur de l'application principale. Elle sert de classe code-behind au fichier MXML de l'application.
Dans le dossier flexapp/filterPanels : BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	Jeu de composants MXML qui assurent la fonctionnalité de chaque panneau utilisé pour définir les options d'un filtre unique.

Fichier	Description
flexapp/ImageContainer.as	Objet d'affichage qui sert de conteneur à l'image chargée à l'écran.
flexapp/controls/BGColorCellRenderer.as	Composant de rendu de cellule personnalisé permettant de modifier la couleur d'arrière-plan d'une cellule dans le composant DataGrid
flexapp/controls/QualityComboBox.as	Contrôle personnalisé qui définit une liste déroulante modifiable associée au paramètre Qualité dans plusieurs panneaux de filtre.
flexapp/controls/TypeComboBox.as	Contrôle personnalisé qui définit une liste déroulante modifiable associée au paramètre Type dans plusieurs panneaux de filtre.
Interface utilisateur (Flash)	
FilterWorkbench.fla	Fichier principal qui définit l'interface utilisateur de l'application.
flashapp/FilterWorkbench.as	Classe qui assure la fonctionnalité de l'interface utilisateur de l'application principale. Elle sert de classe de document au fichier FLA de l'application.
Dans le dossier flashapp/filterPanels : BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	Jeu de classes qui assurent la fonctionnalité de chaque panneau utilisé pour définir les options d'un filtre unique. A chaque classe correspond également un symbole MovieClip dans la bibliothèque du fichier FLA de l'application principale, dont le nom est identique à celui de la classe (par exemple, le symbole « BlurPanel » est lié à la classe définie dans BlurPanel.as). Les composants de l'interface utilisateur sont placés et identifiés par nom dans ces symboles.
flashapp/ImageContainer.as	Objet d'affichage qui sert de conteneur à l'image chargée à l'écran.
flashapp/BGColorCellRenderer.as	Composant de rendu de cellule personnalisé permettant de modifier la couleur d'arrière-plan d'une cellule dans le composant DataGrid
flashapp/ButtonCellRenderer.as	Composant de rendu de cellule personnalisé permettant d'insérer un composant Button dans une cellule du composant DataGrid
Contenu d'image filtré	
com/example/programmingas3/filterWorkbench/ImageType.as	Cette classe sert d'objet de valeur contenant le type et l'URL d'un fichier d'image unique, dans lequel l'application peut charger et appliquer des filtres. Cette classe comporte également un jeu de constantes qui représentent les fichiers d'image en tant que tels disponibles.
images/sampleAnimation.swf, images/sampleImage1.jpg, images/sampleImage2.jpg	Images et autre contenu visuel auxquels sont appliqués des filtres dans l'application.

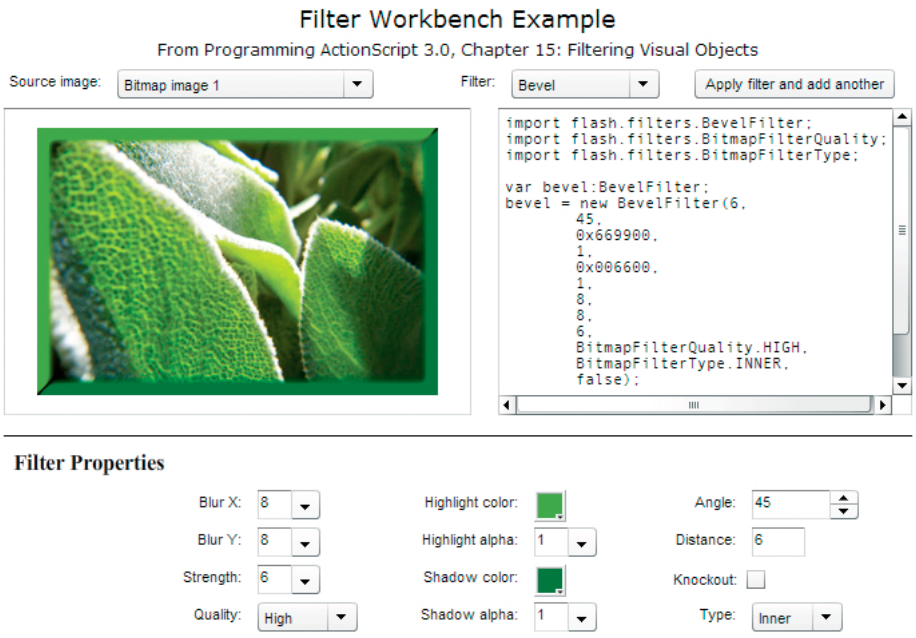
Utilisation des filtres ActionScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application Filter Workbench est conçue pour vous aider à expérimenter avec divers effets de filtre et générer le code ActionScript approprié correspondant. Elle vous permet de sélectionner trois fichiers distincts comportant un contenu visuel, tel que des images bitmap et une animation créée par Flash et d'appliquer huit filtres ActionScript distincts à l'image sélectionnée, soit seuls, soit combinés à d'autres filtres. L'application comprend les filtres suivants :

- Biseau (flash.filters.BevelFilter)
- Flou (flash.filters.BlurFilter)
- Matrice de couleurs (flash.filters.ColorMatrixFilter)
- Convolution (flash.filters.ConvolutionFilter)
- Ombre portée (flash.filters.DropShadowFilter)
- Rayonnement (flash.filters.GlowFilter)
- Biseau dégradé (flash.filters.GradientBevelFilter)
- Rayonnement dégradé (flash.filters.GradientGlowFilter)

Lorsqu'un utilisateur a sélectionné une image et un filtre à appliquer à celle-ci, l'application affiche un panneau contenant des contrôles de définition des propriétés du filtre sélectionné. Par exemple, l'image suivante illustre l'application dans laquelle est sélectionné le filtre Biseau :



Lorsque l'utilisateur règle les propriétés du filtre, l'aperçu est actualisé en temps réel. L'utilisateur peut également appliquer plusieurs filtres. Pour ce faire, il personnalise un filtre, clique sur le bouton Apply, personnalise un autre filtre, clique sur le bouton Apply, et ainsi de suite.

Les panneaux de filtre de l'application proposent diverses fonctions et sont soumis à quelques limites :

- Le filtre Matrice de couleurs comprend un jeu de contrôles permettant de manipuler directement des propriétés d'image courantes telles que la luminosité, les contrastes, la saturation et la teinte. Il est également possible de définir des valeurs de matrice de couleurs personnalisées.
- Le filtre Convolution, qui n'est disponible qu'en ActionScript, comprend un jeu de valeurs de matrice de convolution couramment utilisées. Il est également possible de définir des valeurs personnalisées. Toutefois, bien que la classe ConvolutionFilter gère une matrice de n'importe quelle taille, l'application Filter Workbench utilise une matrice de 3 x 3 fixe, soit la taille de filtre la plus fréquemment utilisée.
- Le filtre Mappage de déplacement et le filtre Shader, réservés à ActionScript, ne sont pas disponibles dans l'application Filter Workbench.

Création d'occurrences de filtre

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application Filter Workbench comprend un jeu de classes, une par filtre disponible, qui sont utilisées par les divers panneaux pour créer les filtres. Lorsqu'un utilisateur sélectionne un filtre, le code ActionScript associé au panneau de filtre crée une occurrence de la classe de filtres usine appropriée. (Ces classes portent le nom de *classes usine*, car elles ont pour objet de créer des occurrences d'autres objets, à l'instar d'une vraie usine qui fabrique des produits).

Lorsque l'utilisateur modifie la valeur d'une propriété dans le panneau, le code correspondant appelle la méthode appropriée dans la classe usine. Chaque classe usine comporte des méthodes spécifiques utilisées par le panneau pour créer l'occurrence de filtre appropriée. Par exemple, si l'utilisateur sélectionne le filtre Flou, l'application crée une occurrence de BlurFactory. La classe BlurFactory comporte une méthode `modifyFilter()` qui gère trois paramètres : `blurX`, `blurY` et `quality`. Conjointement, ces paramètres permettent de créer l'occurrence de BlurFilter requise :

```
private var _filter:BlurFilter;

public function modifyFilter(blurX:Number = 4, blurY:Number = 4, quality:int = 1):void
{
    _filter = new BlurFilter(blurX, blurY, quality);
    dispatchEvent(new Event(Event.CHANGE));
}
```

En revanche, si l'utilisateur sélectionne le filtre Convolution, celui-ci étant beaucoup plus souple, il contrôle un nombre supérieur de propriétés. Dans la classe ConvolutionFactory, le code suivant est appelé lorsque l'utilisateur sélectionne une autre valeur dans le panneau des filtres :

```
private var _filter:ConvolutionFilter;

public function modifyFilter(matrixX:Number = 0,
                                                matrixY:Number = 0,
                                                matrix:Array = null,
                                                divisor:Number = 1.0,
                                                bias:Number = 0.0,
                                                preserveAlpha:Boolean = true,
                                                clamp:Boolean = true,
                                                color:uint = 0,
                                                alpha:Number = 0.0):void
{
    _filter = new ConvolutionFilter(matrixX, matrixY, matrix, divisor, bias, preserveAlpha,
clamp, color, alpha);
    dispatchEvent(new Event(Event.CHANGE));
}
```

Notez que dans chaque cas, lorsque les valeurs du filtre sont modifiées, l'objet usine distribue un événement `Event.CHANGE` pour avertir les écouteurs de la modification. La classe FilterWorkbenchController, qui applique les filtres au contenu filtré, recherche cet événement pour s'assurer qu'elle doit extraire une nouvelle copie du filtre et l'appliquer à nouveau au contenu filtré.

La classe FilterWorkbenchController n'a pas besoin de connaître les détails précis de chaque classe usine associé au filtre. Elle doit juste savoir que le filtre a été modifié et être en mesure d'accéder à une copie de ce dernier. A cet effet, l'application comporte une interface, IFilterFactory, qui définit le comportement requis d'une classe usine de filtres pour que l'occurrence de FilterWorkbenchController de l'application soit en mesure de fonctionner correctement. L'interface IFilterFactory définit la méthode `getFilter()` utilisée par la classe FilterWorkbenchController :

```
function getFilter():BitmapFilter;
```

Notez que la définition de la méthode de l'interface `getFilter()` stipule de renvoyer une occurrence de BitmapFilter plutôt qu'un type déterminé de filtre. La classe BitmapFilter ne définit pas de type spécifique de filtre. BitmapFilter constitue de fait la classe de base sur laquelle se fondent toutes les classes de filtre. Chaque classe usine de filtres définit une implémentation déterminée de la méthode `getFilter()`, dans laquelle elle renvoie une référence à l'objet filtre généré. Par exemple, une version abrégée du code source de la classe ConvolutionFactory est illustrée ci-après :

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

Dans l'implémentation de la classe ConvolutionFactory de la méthode `getFilter()`, elle renvoie une occurrence de ConvolutionFilter, bien que tout objet qui appelle `getFilter()` ne sache pas nécessairement que, conformément à la définition de la méthode `getFilter()` appliquée par ConvolutionFactory, il doit renvoyer toute occurrence de BitmapFilter, soit une occurrence de n'importe quelle classe de filtre ActionScript.

Application de filtres aux objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme indiqué précédemment, l'application Filter Workbench utilise une occurrence de la classe FilterWorkbenchController (appelée à partir de maintenant l'« occurrence de contrôleur »), chargée d'appliquer les filtres à l'objet visuel sélectionné. Avant que l'occurrence de contrôleur ne puisse appliquer un filtre, elle doit identifier l'image ou le contenu visuel concerné. Lorsque l'utilisateur sélectionne une image, l'application appelle la méthode `setFilterTarget()` de la classe FilterWorkbenchController et transmet l'une des constantes définies dans la classe ImageType :

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

En se référant à ces informations, l'occurrence de contrôleur charge le fichier indiqué, puis le stocke dans une variable d'occurrence appelée `_currentTarget` :

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

Lorsque l'utilisateur sélectionne un filtre, l'application appelle la méthode `setFilter()` de l'occurrence de contrôleur en indiquant au contrôleur une référence à l'objet Filter Factory approprié, qu'elle stocke dans une variable d'occurrence appelée `_filterFactory`.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

Notez que, comme indiqué précédemment, l'occurrence de contrôleur ne connaît pas le type de données précis de l'occurrence de Filter Factory assigné. Elle ne sait qu'une chose, c'est que l'objet implémente l'occurrence de IFilterFactory, ce qui signifie qu'il possède une méthode `getFilter()` et distribue un événement `change` (`Event.CHANGE`) lorsque le filtre est modifié.

Lorsque l'utilisateur modifie les propriétés d'un filtre dans le panneau des filtres, l'occurrence de contrôleur est avertie de la modification par l'événement `change` de Filter Factory, qui appelle la méthode `filterChange()` de l'occurrence de contrôleur. Cette méthode appelle alors la méthode `applyTemporaryFilter()` :

```
private function filterChange(event:Event):void
{
    applyTemporaryFilter();
}

private function applyTemporaryFilter():void
{
    var currentFilter:BitmapFilter = _filterFactory.getFilter();

    // Add the current filter to the set temporarily
    _currentFilters.push(currentFilter);

    // Refresh the filter set of the filter target
    _currentTarget.filters = _currentFilters;

    // Remove the current filter from the set
    // (This doesn't remove it from the filter target, since
    // the target uses a copy of the filters array internally.)
    _currentFilters.pop();
}
```

L'application du filtre à l'objet d'affichage se produit au sein de la méthode `applyTemporaryFilter()`. Le contrôleur extrait d'abord une référence à l'objet filtre en appelant la méthode `getFilter()` de Filter Factory.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

L'occurrence de contrôleur possède une variable d'occurrence de Array appelée `_currentFilters`, dans laquelle elle stocke tous les filtres appliqués à l'objet d'affichage. L'étape suivante consiste à ajouter le filtre mis à jour à ce tableau :

```
_currentFilters.push(currentFilter);
```

Le code assigne ensuite le tableau de filtres à la propriété `filters` de l'objet d'affichage, qui applique à proprement parler les filtres à l'image :

```
_currentTarget.filters = _currentFilters;
```

Enfin, puisque le filtre appliqué en dernier demeure le filtre de « travail », il ne doit pas être appliqué à titre définitif à l'objet d'affichage. Il est donc supprimé du tableau `_currentFilters` :

```
_currentFilters.pop();
```

Supprimer ce filtre du tableau n'affecte pas l'objet d'affichage filtré, car un objet d'affichage effectue une copie du tableau de filtres lorsqu'il est assigné à la propriété `filters` et utilise ce tableau interne au lieu du tableau initial. De ce fait, toute modification du tableau de filtres n'affecte pas l'objet d'affichage tant que le tableau n'a pas été à nouveau assigné à la propriété `filters` de l'objet d'affichage.

Chapitre 15 : Utilisation des shaders de Pixel Bender

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Adobe Pixel Bender Toolkit permet aux développeurs de composer des shaders qui créent des effets graphiques et autres. Le pseudo-code binaire Pixel Bender peut être exécuté dans ActionScript pour appliquer l'effet aux données de l'image ou au contenu visuel. L'utilisation des shaders de Pixel Bender vous donne la possibilité de créer des effets visuels personnalisés et de traiter des données au-delà des fonctions incorporées à ActionScript.

Remarque : *Pixel Bender est pris en charge depuis Flash Player 10 et Adobe AIR 1.5. Les fusions, filtres et remplissages Pixel Bender ne sont pas pris en charge en cas de rendu par le processeur graphique.*

Voir aussi

Adobe Pixel Bender Technology Center

Pixel Bender Developer's Guide (disponible en anglais uniquement)

Pixel Bender Reference (disponible en anglais uniquement)

flash.display.Shader

flash.filters.ShaderFilter

Pixel Bender basics for Flash (disponible en anglais uniquement)

Pixel Bender basics for Flex (disponible en anglais uniquement)

Principes de base des shaders de Pixel Bender

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Adobe Pixel Bender est un langage de programmation qui permet de créer ou de manipuler un contenu d'image. Par le biais de Pixel Bender, vous créez un noyau, également appelé un shader. Le shader définit une fonction unique exécutée séparément sur chacun des pixels d'une image. Le résultat de chaque appel à la fonction est la couleur de sortie aux coordonnées de ce pixel dans l'image. Vous pouvez spécifier des valeurs de paramètre et images d'entrée pour personnaliser l'opération. Si un shader est exécuté une seule fois, les valeurs de paramètres et d'entrée sont des constantes. Les seuls éléments qui varient sont les coordonnées du pixel dont la couleur est le résultat de l'appel de la fonction.

Dans la mesure du possible, la fonction shader est appelée pour plusieurs coordonnées de pixel de sortie en parallèle. Les performances du shader sont ainsi optimisées et le traitement s'avère souvent particulièrement efficace.

ActionScript permet de créer facilement trois types d'effets par le biais d'un shader :

- remplissage d'un dessin
- mode de fondu
- filtre

Il est également possible d'exécuter un shader en mode autonome. En mode autonome, vous accédez directement au résultat d'un shader au lieu de prédéfinir l'usage prévu. Le résultat correspond à des données d'image, des données binaires ou des données numériques. Il n'est pas nécessaire que les données soient des données d'image. Vous pouvez ainsi affecter à un shader un jeu de données en entrée. Le shader traite les données et vous accédez au résultat qu'il renvoie.

Pixel Bender est pris en charge depuis Flash Player version 10 et Adobe AIR version 1.5. Les fusions, filtres et remplissages Pixel Bender ne sont pas pris en charge en cas de rendu par processeur graphique. Sur un périphérique mobile, les shaders Pixel Bender sont exécutés en cas de rendu par unité centrale. Les performances ne sont toutefois pas à la hauteur de celles d'un poste de travail. De nombreux programmes de shader ne s'exécutent qu'à quelques images par seconde.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la création et à l'utilisation de shaders de Pixel Bender :

Noyau Pour Pixel Bender, un noyau est équivalent à un shader. Par le biais de Pixel Bender, votre code définit un noyau, qui définit une fonction unique exécutée séparément sur chacun des pixels d'une image.

Pseudo-code binaire Pixel Bender Lorsqu'un noyau Pixel Bender est compilé, il est transformé en pseudo-code binaire Pixel Bender. L'accès au pseudo-code binaire et son exécution se produisent lors de l'exécution.

Langage Pixel Bender Langage de programmation utilisé pour créer un noyau Pixel Bender.

Pixel Bender Toolkit Application utilisée pour créer un fichier de pseudo-code binaire Pixel Bender à partir du code source Pixel Bender. Elle vous permet d'écrire, de tester et de compiler un code source Pixel Bender.

Shader Dans le cadre de ce document, un shader est un jeu de fonctionnalités écrites dans le langage Pixel Bender. Le code d'un shader crée un effet visuel ou effectue un calcul. Dans les deux cas, le shader renvoie un jeu de données (il s'agit en règle générale des pixels d'une image). Le shader exécute la même opération sur chaque point de données, à l'exception des coordonnées du pixel de sortie. Le shader n'est pas écrit en ActionScript. Il est écrit dans le langage Pixel Bender et compilé en pseudo-code binaire Pixel Bender. Il peut être intégré à un fichier SWF lors de la compilation ou chargé en tant que fichier externe lors de l'exécution. Dans les deux cas, pour y accéder dans ActionScript, il est nécessaire de créer un objet Shader et de le lier au pseudo-code binaire du shader.

Entrée du shader Entrée complexe, généralement composée de données d'image bitmap, fournie à un shader qui l'utilise dans ses calculs. Pour chaque variable d'entrée définie dans un shader, une valeur unique (une image unique ou un jeu de données binaires) est utilisée pour l'exécution de bout en bout du shader.

Paramètre de shader Valeur unique (ou jeu de valeurs limité) fournie à un shader, qui l'utilise dans ses calculs. Chaque valeur de paramètre est définie pour une exécution de shader unique et la même valeur est utilisée de bout en bout lors de l'exécution du shader.

Utilisation des exemples de code

Il peut s'avérer utile de tester les exemples de code fournis. Cette opération nécessite d'exécuter le code et d'afficher les résultats dans le fichier SWF créé. Tous les exemples créent un contenu par le biais de l'API de dessin qui utilise l'effet de shader ou est modifiée par ce dernier.

La plupart des exemples de code se composent de deux parties. La première partie correspond au code source Pixel Bender associé au shader utilisé dans l'exemple. Vous devez commencer par utiliser Pixel Bender Toolkit pour compiler le code source dans un fichier de pseudo-code binaire Pixel Bender. Procédez comme suit pour créer le fichier de pseudo-code binaire Pixel Bender :

1 Ouvrez Adobe Pixel Bender Toolkit. Le cas échéant, dans le menu Build (Développement), choisissez « Turn on Flash Player warnings and errors » (Activer les avertissements et erreurs Flash Player).

2 Copiez le code Pixel Bender et collez-le dans le panneau de l'éditeur de code de Pixel Bender Toolkit.

3 Dans le menu File (Fichier), choisissez « Export kernel filter for Flash Player » (Exporter le filtre de noyau associé à Flash Player).

4 Enregistrez le fichier de pseudo-code binaire Pixel Bender dans le même répertoire que le document Flash. Le nom du fichier doit être identique à celui stipulé dans la description de l'exemple.

La partie ActionScript de chaque exemple est écrite en tant que fichier de classe. Pour tester l'exemple dans Flash Professional :

1 Créez un document Flash vide et enregistrez-le sur votre ordinateur.

2 Créez un nouveau fichier ActionScript et enregistrez-le dans le même répertoire que le document Flash. Le nom du fichier doit correspondre au nom de la classe du code. Par exemple, si le code définit une classe MyApplication, enregistrez le fichier ActionScript sous le nom MyApplication.as.

3 Copiez le code dans le fichier ActionScript et enregistrez le fichier.

4 Dans le document Flash, cliquez sur une partie vide de la scène ou de l'espace de travail pour activer l'Inspecteur des propriétés du document.

5 Dans l'Inspecteur des propriétés, dans le champ Classe du document, saisissez le nom de la classe ActionScript que vous avez copiée du texte.

6 Exécutez le programme en sélectionnant Contrôle > Tester l'animation.

Les résultats de l'exemple s'affichent dans la fenêtre d'aperçu.

Ces techniques de test des exemples de code font l'objet d'une description détaillée dans « Utilisation des exemples de code ActionScript » à la page 1145.

Chargement ou intégration d'un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La première étape dans l'utilisation d'un shader de Pixel Bender dans ActionScript consiste à pouvoir accéder au shader en code ActionScript. Comme un shader est créé à l'aide d'Adobe Pixel Bender Toolkit et rédigé dans le langage Pixel Bender, on ne peut pas y accéder directement dans ActionScript. Il faut plutôt créer une occurrence de la classe Shader qui représente le shader de Pixel Bender à ActionScript. L'objet Shader vous permet de trouver les informations concernant le shader : par exemple, s'il prévoit de recevoir des paramètres ou des valeurs pour l'image d'entrée. Vous passez l'objet Shader aux autres objets pour utiliser effectivement le shader. Par exemple, pour utiliser le shader comme filtre, vous affectez l'objet Shader à la propriété `shader` d'un objet ShaderFilter. Ou bien, afin d'utiliser le shader pour remplir l'écran dans un dessin, vous passez l'objet Shader comme argument à la méthode `Graphics.beginShaderFill()`.

Votre code ActionScript peut accéder à un shader créé par Adobe Pixel Bender Toolkit (un fichier .pbj) de deux façons distinctes :

- Chargement lors de l'exécution : le fichier shader peut être chargé comme un élément externe à l'aide d'un objet URLLoader. Cette technique est comparable au chargement d'un élément externe, comme un fichier texte, par exemple. L'exemple suivant montre le chargement à l'exécution d'un fichier de pseudo-code binaire de shader et sa liaison à une occurrence de Shader :

```
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.BINARY;
loader.addEventListener(Event.COMPLETE, onLoadComplete);
loader.load(new URLRequest("myShader.pbj"));

var shader:Shader;

function onLoadComplete(event:Event):void {
    // Create a new shader and set the loaded data as its bytecode
    shader = new Shader();
    shader.byteCode = loader.data;

    // You can also pass the bytecode to the Shader() constructor like this:
    // shader = new Shader(loader.data);

     // do something with the shader
}
```

- Intégration au fichier SWF : le fichier shader peut être intégré au fichier SWF lors de la compilation à l'aide de la balise de métadonnées `[Embed]`. La balise de métadonnées `[Embed]` n'est disponible que si vous utilisez le kit de développement Flex pour compiler le fichier SWF. Le paramètre `source` de la balise `[Embed]` pointe vers le fichier du shader et son paramètre `mimeType` est `"application/octet-stream"`, comme dans l'exemple ci-dessous :

```
[Embed(source="myShader.pbj", mimeType="application/octet-stream")]
var MyShaderClass:Class;

// ...

// create a shader and set the embedded shader as its bytecode
var shader:Shader = new Shader();
shader.byteCode = new MyShaderClass();

// You can also pass the bytecode to the Shader() constructor like this:
// var shader:Shader = new Shader(new MyShaderClass());

// do something with the shader
```

Dans les deux cas, vous liez le pseudo-code brut du shader (la propriété `URLLoader.data` ou une occurrence de la classe de données `[Embed]`) à l'occurrence de Shader. Comme le montrent les exemples précédents, vous pouvez affecter le pseudo-code binaire à l'occurrence du shader de deux façons. Vous pouvez transmettre le pseudo-code binaire du shader sous forme d'argument au constructeur `Shader()`. Vous pouvez également le définir en tant que propriété `byteCode` de l'occurrence de Shader.

Dès qu'un shader de Pixel Bender est créé et lié à l'objet Shader, vous pouvez utiliser le shader pour créer des effets de plusieurs façons. Vous pouvez l'utiliser en tant que filtre, mode de fondu, remplissage de bitmap ou comme moyen de traitement autonome d'image bitmap ou autres données. Vous pouvez également utiliser la propriété `data` de l'objet Shader pour accéder aux métadonnées du shader, spécifier les images d'entrée et spécifier les valeurs du paramétrage.

Accès aux métadonnées du shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Tandis qu'il crée un noyau de shader de Pixel Bender, le programmeur peut spécifier des métadonnées sur le shader dans le code source Pixel Bender. Vous pouvez inspecter le shader et extraire ses métadonnées pendant que vous l'utilisez dans ActionScript.

Lorsque vous créez une occurrence de Shader et que vous la liez à un shader de Pixel Bender, un objet ShaderData, qui contient des données sur le shader, est créé et enregistré dans la propriété `data` de l'objet Shader. La classe ShaderData ne définit aucune de ses propres propriétés. Toutefois, lors de l'exécution, une propriété est ajoutée dynamiquement à l'objet ShaderData pour chaque valeur de métadonnées définie dans le code source du shader. Le nom attribué à chaque propriété est pareil à celui spécifié dans les métadonnées. Par exemple, supposez que le code source d'un shader de Pixel Bender contienne la définition des métadonnées suivante :

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

L'objet ShaderData créé pour ce shader l'est avec les propriétés et valeurs suivantes :

- `namespace` (String): `"Adobe::Example"`
- `vendor` (String): `"Bob Jones"`
- `version` (String): `"1"`
- `description` (String): `"Creates a version of the specified image with the specified brightness"`

Comme les propriétés de métadonnées sont ajoutées dynamiquement à l'objet ShaderData, vous pouvez utiliser une boucle `for..in` pour inspecter l'objet ShaderData. Cette technique vous permet de vous rendre compte si le shader possède des métadonnées et quelles sont ses valeurs. Outre les propriétés des métadonnées, un objet ShaderData peut avoir des propriétés représentant des entrées et des paramètres qui sont définis dans le shader. Lorsque vous utilisez une boucle `for..in` pour inspecter un objet ShaderData, vérifiez le type de données de chaque propriété pour vous rendre compte si la propriété est une entrée (une occurrence de ShaderInput), un paramètre (une occurrence de ShaderParameter) ou une valeur de métadonnées (une occurrence de String). L'exemple ci-dessous montre comment utiliser une boucle `for..in` pour examiner les propriétés dynamiques de la propriété `data` d'un shader. Chaque valeur de métadonnées est ajoutée à une occurrence de Vector appelée `metadata`. Remarquez que cet exemple suppose qu'une occurrence de Shader appelée `myShader` existe déjà :

```
var shaderData:ShaderData = myShader.data;
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in shaderData)
{
    if (!(shaderData[prop] is ShaderInput) && !(shaderData[prop] is ShaderParameter))
    {
        metadata[metadata.length] = shaderData[prop];
    }
}

// do something with the metadata
```

Pour une version de cet exemple, qui extrait également des entrées et des paramètres de shader, voir « Identification des entrées et des paramètres d'un shader » à la page 315. Pour plus d'informations sur les propriétés des entrées et des paramètres, voir « Spécification des valeurs des entrées et des paramètres d'un shader » à la page 315.

Spécification des valeurs des entrées et des paramètres d'un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

De nombreux shaders de Pixel Bender sont définis pour faire appel à une ou plusieurs images utilisées au cours de son traitement. Par exemple, il est courant pour un shader d'accepter une image source et de la produire dotée d'un effet particulier. Suivant l'utilisation du shader, soit la valeur d'entrée est spécifiée automatiquement, soit il faut en fournir une. De la même façon, de nombreux shaders spécifient des paramètres utilisés dans l'individualisation de ce qu'ils produisent. Il faut également définir explicitement une valeur pour chaque paramètre avant d'utiliser le shader.

Vous utilisez la propriété `data` de l'objet Shader pour définir les entrées et les paramètres et pour établir si un shader en particulier prévoit des entrées et des paramètres. La propriété `data` est une occurrence de ShaderData.

Identification des entrées et des paramètres d'un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La première étape dans la spécification de valeurs d'entrée et de paramètres d'un shader consiste à savoir si celui que vous utilisez prévoit de recevoir des images d'entrée ou des paramètres. Chaque occurrence de Shader possède une propriété `data` qui contient un objet ShaderData. Si le shader spécifie des entrées ou des paramètres, on y accède en tant que propriétés de cet objet ShaderData. Les noms des propriétés correspondent aux noms spécifiés dans le code source du shader pour les entrées et les paramètres. Par exemple, si un shader spécifie une entrée appelée `src`, l'objet ShaderData possède une propriété appelée `src` qui représente cette entrée. Chaque propriété qui représente une entrée est une occurrence de ShaderInput et chaque propriété qui représente un paramètre est une occurrence de ShaderParameter.

Idéalement, le programmeur du shader fournit une documentation pour le shader afin de décrire quels sont les paramètres et les valeurs de l'image d'entrée qu'il prévoit, ce qu'ils représentent, les valeurs appropriées, et ainsi de suite.

Toutefois, si le shader n'est pas documenté et que vous ne disposez pas du code source, vous pouvez inspecter les données du shader pour identifier ces entrées et paramètres. Les propriétés qui représentent ces entrées et paramètres sont ajoutées dynamiquement à l'objet ShaderData. Par conséquent, vous pouvez utiliser une boucle `for..in` pour inspecter l'objet ShaderData afin de savoir si le shader qui lui est associé spécifie de tels entrées ou paramètres. Comme le décrit la section « Accès aux métadonnées du shader » à la page 314, on accède également, en tant que propriété dynamique ajoutée à la propriété `Shader.data`, à toute valeur de métadonnées définie pour un shader. Lorsque vous utilisez cette technique afin d'identifier entrées et paramètres, vérifiez le type de données des propriétés dynamiques. Si une propriété est une occurrence de ShaderInput, elle représente une entrée. S'il s'agit d'une occurrence de ShaderParameter, elle représente un paramètre. Sinon, il s'agit d'une valeur de métadonnées. L'exemple ci-dessous montre comment utiliser une boucle `for..in` pour examiner les propriétés dynamiques de la propriété `data` d'un shader. Chaque entrée (objet ShaderInput) est ajoutée à une occurrence de Vector appelée `inputs`. Chaque paramètre (objet ShaderParameter) est ajouté à une occurrence de Vector appelée `parameters`. Enfin, toute propriété de métadonnées est ajoutée à une occurrence de Vector appelée `metadata`. Remarquez que cet exemple suppose qu'une occurrence de Shader appelée `myShader` existe déjà :

```
var shaderData:ShaderData = myShader.data;
var inputs:Vector.<ShaderInput> = new Vector.<ShaderInput>();
var parameters:Vector.<ShaderParameter> = new Vector.<ShaderParameter>();
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in shaderData)
{
    if (shaderData[prop] is ShaderInput)
    {
        inputs[inputs.length] = shaderData[prop];
    }
    else if (shaderData[prop] is ShaderParameter)
    {
        parameters[parameters.length] = shaderData[prop];
    }
    else
    {
        metadata[metadata.length] = shaderData[prop];
    }
}

// do something with the inputs or properties
```

Spécification des valeurs d'entrée d'un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

De nombreux shaders prévoient de recevoir une ou plusieurs images d'entrée qui sont utilisées au cours de son traitement. Toutefois, dans nombre de cas, une entrée est automatiquement spécifiée lorsque l'objet Shader est utilisé. Par exemple, supposons qu'un shader ait besoin d'une entrée et qu'il serve de filtre. Lorsque le filtre est appliqué à un objet d'affichage ou BitmapData, cet objet est défini automatiquement comme entrée. Dans ce cas, on ne spécifie pas explicitement une valeur d'entrée.

Toutefois, dans certains cas, et plus particulièrement si un shader spécifie plusieurs entrées, il faut définir explicitement une valeur pour l'entrée. Toute entrée définie dans un shader est représentée dans ActionScript par un objet ShaderInput. L'objet ShaderInput est une propriété de l'occurrence de ShaderData dans la propriété `data` de l'objet Shader (voir « Identification des entrées et des paramètres d'un shader » à la page 315). Par exemple, supposons qu'un shader définisse une entrée appelée `src` et qu'il soit lié à un objet Shader appelé `myShader`. Vous accédez alors à l'objet ShaderInput qui correspond à l'entrée `src` à l'aide de l'identifiant suivant :

```
myShader.data.src
```

Chaque objet ShaderInput possède une propriété `input` qui est utilisée pour définir la valeur de l'entrée. On définit la propriété `input` sur une occurrence de BitmapData pour spécifier des données d'image. On peut aussi définir la propriété `input` sur BitmapData ou Vector.occurrence de <Number> pour spécifier des données binaires ou des nombres. Pour plus d'informations sur l'utilisation d'une occurrence de BitmapData ou Vector.<Number> en entrée et les restrictions correspondantes, voir `ShaderInput.input` dans le Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

En plus de la propriété `input`, un objet ShaderInput possède des propriétés qui peuvent être utilisées pour déterminer quel type d'image prévoit l'entrée. Ces propriétés contiennent elles-mêmes les propriétés `width`, `height` et `channels`. Chaque objet ShaderInput possède également une propriété `index` qui est utile afin de déterminer si une valeur explicite doit être fournie pour l'entrée. Si un shader prévoit de recevoir davantage d'entrées que le nombre fixé automatiquement, vous pouvez alors définir des valeurs pour ces entrées. Pour plus de détails sur les différentes façons d'utiliser un shader et pour savoir si oui ou non les valeurs d'entrée sont fixées automatiquement, voir « Utilisation d'un shader » à la page 320.

Spécification des valeurs de paramètres dans un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Certains shaders définissent des valeurs de paramètres qu'ils utilisent dans la création de leur résultat. Par exemple, un shader qui modifie la luminosité d'une image pourrait spécifier un paramètre de luminosité qui détermine dans quelle mesure l'opération affecte cette luminosité. Un paramètre unique défini dans un shader peut prévoir une ou plusieurs valeurs suivant la façon dont il a été spécifié dans le shader. Tout paramètre défini dans un shader est représenté dans ActionScript par un objet ShaderParameter. L'objet ShaderParameter est une propriété de l'occurrence de ShaderData dans la propriété des données de l'objet Shader. Une description se trouve dans la section « Identification des entrées et des paramètres d'un shader » à la page 315. Par exemple, supposons qu'un shader qui définit un paramètre appelé `luminosité` soit représenté par un objet Shader appelé `myShader`. Vous accédez alors à l'objet ShaderParameter correspondant au paramètre `luminosité` à l'aide de l'identifiant suivant :

```
myShader.data.brightness
```

Pour spécifier une ou plusieurs valeurs pour le paramètre, créez un tableau ActionScript contenant la ou les valeurs et affectez-le à la propriété `value` de l'objet ShaderParameter. La propriété `value` est définie en tant qu'occurrence d'Array car il est possible qu'un seul paramètre du shader nécessite plusieurs valeurs. Même si le paramètre du shader ne prévoit qu'une seule valeur, il vous faut placer la valeur dans un objet Array pour l'affecter à la propriété `ShaderParameter.value`. Le code suivant montre comment définir une seule valeur pour la propriété `value`.

```
myShader.data.brightness.value = [75];
```

Si le code source Pixel Bender pour le shader spécifie une valeur par défaut pour le paramètre, un tableau contenant la ou les valeurs par défaut est créé et affecté à la propriété `value` de l'objet ShaderParameter lorsque l'objet Shader est créé. Dès qu'un tableau est affecté à la propriété `value`, même s'il s'agit de celui par défaut, la valeur du paramètre peut être modifiée en changeant la valeur de l'élément du tableau. Il n'est pas nécessaire de créer un autre tableau et de l'affecter à la propriété `value`.

L'exemple ci-dessous indique comment définir une valeur de paramètre dans un shader avec ActionScript. Dans cet exemple, le shader définit un paramètre appelé `color`. Ce paramètre `color` est déclaré en tant que variable `float4` dans le code source Pixel Bender, ce qui signifie qu'il s'agit d'un tableau à quatre nombres en virgule flottante. Dans l'exemple, la valeur du paramètre `color` change constamment. A chaque changement, le shader est utilisé pour dessiner un rectangle coloré à l'écran. Il en résulte un changement de couleur animé.

Remarque : *le code de cet exemple a été rédigé par Ryan Taylor. Merci Ryan de bien vouloir nous en faire profiter. Vous pouvez accéder aux exemples de Ryan et lire ses commentaires à l'adresse www.boostworthy.com/.*

Le code ActionScript repose sur l'utilisation de trois méthodes :

- `init()` : dans la méthode `init()`, le code charge le fichier de pseudo-code binaire Pixel Bender contenant le shader. Lors du chargement du fichier, la méthode `onLoadComplete()` est appelée.
- `onLoadComplete()` : dans la méthode `onLoadComplete()`, le code crée l'objet Shader appelé `shader`. Il crée également une occurrence de Sprite appelée `texture`. Dans la méthode `renderShader()`, le code dessine une fois par image le résultat du shader dans `texture`.

- `onEnterFrame()` : la méthode `onEnterFrame()` est appelée une fois par image, créant ainsi un effet d'animation. Dans cette méthode, le code définit la valeur du paramètre du shader sur la nouvelle couleur, puis appelle la méthode `renderShader()` pour dessiner le résultat du shader sous forme de rectangle.
- `renderShader()` : dans la méthode `renderShader()`, le code appelle la méthode `Graphics.beginShaderFill()` pour définir un remplissage de shader. Il dessine ensuite un rectangle dont le remplissage est défini par la sortie du shader (la couleur générée). Pour plus d'informations sur ce type d'utilisation d'un shader, voir « Utilisation d'un shader comme outil de remplissage de dessin » à la page 320.

Vous trouverez ci-dessous le code ActionScript associé à cet exemple : Utilisez cette classe en tant que classe d'application principale d'un projet ActionScript uniquement dans Flash Builder, ou en tant que classe de document du fichier FLA dans Flash Professional :

```
package
{
    import flash.display.Shader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class ColorFilterExample extends Sprite
    {
        private const DELTA_OFFSET:Number = Math.PI * 0.5;
        private var loader:URLLoader;
        private var shader:Shader;
        private var texture:Sprite;
        private var delta:Number = 0;

        public function ColorFilterExample()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("ColorFilter.pbj"));
        }

        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            texture = new Sprite();

            addChild(texture);

            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
        private function onEnterFrame(event:Event):void
        {
```

```
            shader.data.color.value[0] = 0.5 + Math.cos(delta - DELTA_OFFSET) * 0.5;
            shader.data.color.value[1] = 0.5 + Math.cos(delta) * 0.5;
            shader.data.color.value[2] = 0.5 + Math.cos(delta + DELTA_OFFSET) * 0.5;
            // The alpha channel value (index 3) is set to 1 by the kernel's default
            // value. This value doesn't need to change.

            delta += 0.1;

            renderShader();
        }

        private function renderShader():void
        {
            texture:graphics.clear();
            texture.graphics.beginShaderFill(shader);
            texture.graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
            texture.graphics.endFill();
        }
    }
}
```

Vous trouverez ci-dessous le code source du noyau du shader ColorFilter, utilisé pour la création du fichier « ColorFilter.pbj » de pseudo-code binaire Pixel Bender.

```
<languageVersion : 1.0;>
kernel ColorFilter
<
    namespace : "boostworthy::Example";
    vendor : "Ryan Taylor";
    version : 1;
    description : "Creates an image where every pixel has the specified color value.";
>
{
    output pixel4 result;

    parameter float4 color
    <
        minValue:float4(0, 0, 0, 0);
        maxValue:float4(1, 1, 1, 1);
        defaultValue:float4(0, 0, 0, 1);
    >;

    void evaluatePixel()
    {
        result = color;
    }
}
```

Si vous utilisez un shader dont les paramètres ne sont pas documentés, c'est par le contrôle de la propriété `type` de l'objet ShaderParameter que vous pouvez évaluer le nombre d'éléments et leur type qu'il faut inclure dans le tableau. La propriété `type` indique le type de données du paramètre tel qu'il est défini dans le shader lui-même. Pour consulter la liste du nombre et du type d'éléments prévus par chaque type de paramètre, voir le code associé à la propriété `ShaderParameter.value` dans le Guide de référence ActionScript 3.0 pour Flash Professional.

Chaque objet ShaderParameter possède également une propriété `index` qui indique l'emplacement du paramètre dans l'ordre des paramètres du shader. En plus de ces propriétés, un objet ShaderParameter peut avoir des propriétés supplémentaires qui contiennent des valeurs de métadonnées fournies par le programmeur du shader. Par exemple, le programmeur peut définir pour un paramètre des valeurs de métadonnées telles que minimum, maximum et des valeurs par défaut. Toutes les valeurs de métadonnées spécifiées par le programmeur sont ajoutées à l'objet ShaderParameter en tant que propriétés dynamiques. Pour passer en revue ces propriétés, utilisez une boucle `for..in` pour boucler dans les propriétés dynamiques de l'objet ShaderParameter afin d'identifier ses métadonnées. L'exemple ci-dessous montre comment utiliser une boucle `for..in` afin d'identifier les métadonnées d'un objet ShaderParameter. Chaque valeur de métadonnées est ajoutée à une occurrence de Vector appelée `metadata`. Remarquez que cet exemple suppose qu'une occurrence de Shader appelée `myShader` existe déjà et qu'elle possède un paramètre appelé `brightness`.

```
var brightness:ShaderParameter = myShader.data.brightness;
var metadata:Vector.<String> = new Vector.<String>();

for (var prop:String in brightness)
{
    if (brightness[prop] is String)
    {
        metadata[metadata.length] = brightness[prop];
    }
}

// do something with the metadata
```

Utilisation d'un shader

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Dès qu'un shader de Pixel Bender est disponible dans ActionScript en tant qu'objet Shader, il peut être utilisé de différentes façons :

- Remplissage d'un dessin par le shader : le shader spécifie quelle portion d'une forme dessinée utilise l'api du dessin
- Mode de fondu : le shader spécifie le degré de fondu entre deux objets d'affichage superposés
- Filtre : le shader définit un filtre qui modifie l'apparence du contenu visuel
- Traitement d'un shader autonome : le traitement du shader se fait sans définir l'usage de la sortie. Sur demande, le shader peut tourner en arrière-plan de telle sorte que le résultat devient disponible à la fin du traitement. Cette technique peut être utilisée pour générer des données bitmap et traiter des données non visuelles.

Utilisation d'un shader comme outil de remplissage de dessin

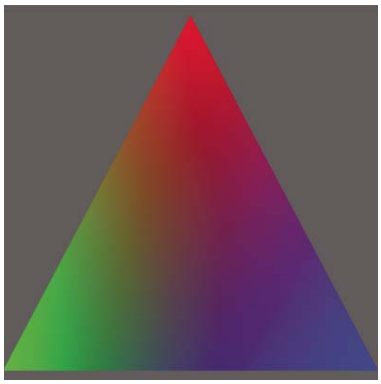
Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Si vous utilisez un shader pour créer un remplissage de dessin, vous faites appel aux méthodes de l'API de dessin pour créer une forme vectorielle. La sortie du shader permet de remplir la forme, à l'instar de toute image bitmap utilisée en tant que remplissage de bitmap par l'API de dessin. Pour créer un remplissage de dessin, appelez la méthode `beginShaderFill()` de l'objet Graphics à l'emplacement du code où vous souhaitez commencer à dessiner la forme. Transmettez l'objet Shader en tant que premier argument à la méthode `beginShaderFill()`, comme illustré dans le code suivant :

```
var canvas:Sprite = new Sprite();
canvas.graphics.beginShaderFill(myShader);
canvas.graphics.drawRect(10, 10, 150, 150);
canvas.graphics.endFill();
// add canvas to the display list to see the result
```

Lorsque vous utilisez un shader en tant qu'outil de remplissage de dessin, vous définissez les valeurs d'image d'entrée et de paramètre requises par le shader.

L'exemple suivant illustre l'utilisation d'un shader en tant qu'outil de remplissage de dessin. Dans cet exemple, le shader crée un dégradé à trois sommets. Ce dégradé possède trois couleurs, une par sommet d'un triangle, séparées des autres par un fondu dégradé. En outre, les couleurs pivotent pour créer un effet de rotation de couleur animé.



Remarque : *le code de cet exemple a été rédigé par Petri Leskinen. Merci Petri de bien vouloir nous en faire profiter. Vous pouvez consulter d'autres exemples et didacticiels rédigés par Petri à l'adresse http://pixelero.wordpress.com/.*

Le code ActionScript réside dans trois méthodes :

- `init()` : la méthode `init()` est appelée lors du chargement de l'application. Dans cette méthode, le code définit les valeurs initiales des objets Point qui représentent les sommets du triangle. Il crée également une occurrence de Sprite appelée `canvas`. Ultérieurement, dans la méthode `updateShaderFill()`, le code dessine une fois par image le résultat du shader dans `canvas`. Enfin, le code charge le fichier de pseudo-code binaire du shader.
- `onLoadComplete()` : dans la méthode `onLoadComplete()`, le code crée l'objet Shader appelé `shader`. Il définit également les valeurs initiales des paramètres. Enfin, le code ajoute la méthode `updateShaderFill()` en tant qu'écouteur de l'événement `enterFrame`, ce qui signifie qu'il est appelé une fois par image pour créer un effet animé.
- `onEnterFrame()` : la méthode `updateShaderFill()` est appelée une fois par image, ce qui crée l'effet d'animation. Dans cette méthode, le code calcule et définit les valeurs des paramètres du shader. Il appelle ensuite la méthode `beginShaderFill()` pour créer un remplissage de shader et appelle d'autres méthodes de l'API de dessin pour dessiner le résultat du shader dans un triangle.

Vous trouverez ci-dessous le code ActionScript associé à cet exemple : Utilisez cette classe en tant que classe d'application principale d'un projet ActionScript uniquement dans Flash Builder, ou en tant que classe de document d'un fichier FLA dans Flash Professional :

```
package
{
    import flash.display.Shader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.geom.Point;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class ThreePointGradient extends Sprite
    {
        private var canvas:Sprite;
        private var shader:Shader;
        private var loader:URLLoader;

        private var topMiddle:Point;
        private var bottomLeft:Point;
        private var bottomRight:Point;

        private var colorAngle:Number = 0.0;
        private const d120:Number = 120 / 180 * Math.PI; // 120 degrees in radians


        public function ThreePointGradient()
        {
            init();
        }

        private function init():void
        {
            canvas = new Sprite();
            addChild(canvas);

            var size:int = 400;
            topMiddle = new Point(size / 2, 10);
            bottomLeft = new Point(0, size - 10);
            bottomRight = new Point(size, size - 10);

            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("ThreePointGradient.pbj"));
        }

        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            shader.data.point1.value = [topMiddle.x, topMiddle.y];
            shader.data.point2.value = [bottomLeft.x, bottomLeft.y];
            shader.data.point3.value = [bottomRight.x, bottomRight.y];

            addEventListener(Event.ENTER_FRAME, updateShaderFill);
        }

        private function updateShaderFill(event:Event):void
```

```
        {
            colorAngle += .06;

            var c1:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle);
            var c2:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle + d120);
            var c3:Number = 1 / 3 + 2 / 3 * Math.cos(colorAngle - d120);

            shader.data.color1.value = [c1, c2, c3, 1.0];
            shader.data.color2.value = [c3, c1, c2, 1.0];
            shader.data.color3.value = [c2, c3, c1, 1.0];

            canvas.graphics.clear();
            canvas.graphics.beginShaderFill(shader);

            canvas.graphics.moveTo(topMiddle.x, topMiddle.y);
            canvas.graphics.lineTo(bottomLeft.x, bottomLeft.y);
            canvas.graphics.lineTo(bottomRight.x, bottomLeft.y);

            canvas.graphics.endFill();
        }
    }
}
```

Le code source du noyau du shader ThreePointGradient, utilisé pour la création du fichier « ThreePointGradient.pbj » de pseudo-code binaire Pixel Bender, est indiqué ci-dessous :

```
<languageVersion : 1.0;>
kernel ThreePointGradient
<
    namespace : "Petri Leskinen::Example";
    vendor : "Petri Leskinen";
    version : 1;
    description : "Creates a gradient fill using three specified points and colors.";
>
{
    parameter float2 point1 // coordinates of the first point
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 0);
    >;

    parameter float4 color1 // color at the first point, opaque red by default
    <
        defaultValue:float4(1.0, 0.0, 0.0, 1.0);
    >;

    parameter float2 point2 // coordinates of the second point
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 500);
    >;

    parameter float4 color2 // color at the second point, opaque green by default
    <
        defaultValue:float4(0.0, 1.0, 0.0, 1.0);
```

```
    >;

    parameter float2 point3 // coordinates of the third point
    <
        minValue:float2(0, 0);
        maxValue:float2(4000, 4000);
        defaultValue:float2(0, 500);
    >;

    parameter float4 color3 // color at the third point, opaque blue by default
    <
        defaultValue:float4(0.0, 0.0, 1.0, 1.0);
    >;

    output pixel4 dst;

    void evaluatePixel()
    {
        float2 d2 = point2 - point1;
        float2 d3 = point3 - point1;

        // transformation to a new coordinate system
        // transforms point 1 to origin, point2 to (1, 0), and point3 to (0, 1)
        float2x2 mtrx = float2x2(d3.y, -d2.y, -d3.x, d2.x) / (d2.x * d3.y - d3.x * d2.y);
        float2 pNew = mtrx * (outCoord() - point1);

        // repeat the edge colors on the outside
        pNew.xy = clamp(pNew.xy, 0.0, 1.0); // set the range to 0.0 ... 1.0

        // interpolating the output color or alpha value
        dst = mix(mix(color1, color2, pNew.x), color3, pNew.y);
    }
}
```

Remarque : *si vous utilisez un remplissage de shader alors que le rendu est exécuté par le processeur graphique, la zone remplie s'affiche en cyan.*

Pour plus d'informations sur le tracé de forme par le biais de l'API de dessin, voir « Utilisation de l'API de dessin » à la page 230.

Utilisation d'un shader comme mode de fondu

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Utiliser un shader comme mode de fondu s'apparente à l'utilisation des autres modes de fondu. Le shader définit le résultat de deux objets d'affichage fondus visuellement. Pour utiliser un shader en tant que mode de fondu, affectez à votre objet Shader la propriété `blendShader` de l'objet d'affichage en premier plan. Affecter une valeur autre que `null` à la propriété `blendShader` définit automatiquement la propriété `blendMode` de l'objet d'affichage sur `BlendMode.SHADER`. Le code suivant illustre l'utilisation d'un shader en tant que mode de fondu. Notez que cet exemple part du principe qu'un objet d'affichage appelé `foreground` figure dans le même parent que l'autre contenu d'affichage dans la liste d'affichage, sachant que `foreground` chevauche l'autre contenu :

```
foreground.blendShader = myShader;
```

Lorsque vous utilisez un shader en tant que mode de fondu, deux entrées au moins doivent être spécifiées. Comme indiqué dans l'exemple, vous ne définissez pas les valeurs d'entrée dans le code. Les deux images fondues sont automatiquement utilisées en tant qu'entrées du shader. L'image en premier-plan fait office de seconde image (c'est à cet objet d'affichage qu'est appliqué le mode de fondu). Une image d'arrière-plan est créée à partir du composite de tous les pixels figurant derrière le cadre de sélection de l'image en premier-plan. Cette image en arrière-plan est définie comme la première image d'entrée. Si vous utilisez un shader qui attend plus de deux entrées, vous en indiquez la valeur à partir de la troisième entrée.

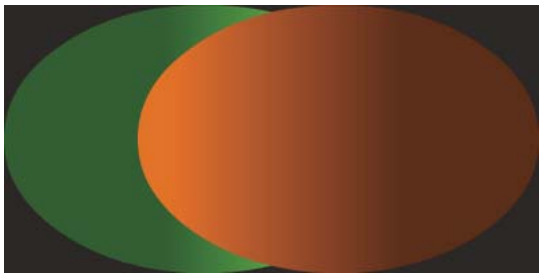
L'exemple suivant illustre l'utilisation d'un shader en tant que mode de fondu. Cet exemple utilise un mode de fondu Eclaircir basé sur la luminosité. Ce fondu a pour résultat d'afficher le pixel le plus clair de l'un ou l'autre des objets fondus.

Remarque : *le code de cet exemple a été rédigé par Mario Klingemann. Merci Mario de bien vouloir nous en faire profiter. Pour consulter d'autres exemples de code rédigés par Mario, ainsi que ses commentaires, voir www.quasimondo.com/.*

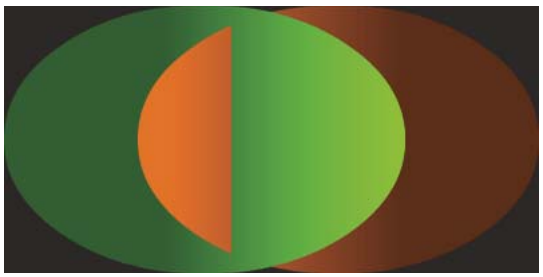
Le code ActionScript important figure dans les deux méthodes suivantes :

- `init()` : la méthode `init()` est appelée lors du chargement de l'application. Dans cette méthode, le code charge le fichier de pseudo-code binaire du shader.
- `onLoadComplete()` : dans la méthode `onLoadComplete()`, le code crée l'objet Shader appelé `shader`. Il dessine ensuite trois objets. Le premier, `backdrop`, est un arrière-plan gris foncé placé derrière les objets fondus. Le deuxième, `backgroundShape`, est une ellipse dégradée verte. Le troisième, `foregroundShape`, est une ellipse dégradée orange.

  L'ellipse `foregroundShape` est l'objet de premier plan du fondu. L'image d'arrière-plan du fondu est composée de la section de `backdrop` et de la section de `backgroundShape` qui sont recouvertes par le cadre de sélection de l'objet `foregroundShape`. L'objet `foregroundShape` est affiché en première position dans la liste d'affichage. Il recouvre partiellement `backgroundShape` et totalement `backdrop`. En raison de ce chevauchement, sans application d'un mode de fondu, l'ellipse orange (`foregroundShape`) est entièrement affichée et masque une section de l'ellipse verte (`backgroundShape`) :



  Toutefois, si le mode de fondu est affiché, la section la plus lumineuse de l'ellipse verte est visible, car elle est plus claire que la section de `foregroundShape` qui la recouvre :



Vous trouverez ci-dessous le code ActionScript associé à cet exemple : Utilisez cette classe en tant que classe d'application principale d'un projet ActionScript uniquement dans Flash Builder, ou en tant que classe de document du fichier FLA dans Flash Professional :

```
package
{
    import flash.display.BlendMode;
    import flash.display.GradientType;
    import flash.display.Graphics;
    import flash.display.Shader;
    import flash.display.Shape;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.geom.Matrix;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class LumaLighten extends Sprite
    {
        private var shader:Shader;
        private var loader:URLLoader;

        public function LumaLighten()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("LumaLighten.pbj"));
        }


        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            var backdrop:Shape = new Shape();
            var g0:Graphics = backdrop.graphics;
            g0.beginFill(0x303030);
            g0.drawRect(0, 0, 400, 200);
            g0.endFill();
            addChild(backdrop);

            var backgroundShape:Shape = new Shape();
            var g1:Graphics = backgroundShape.graphics;
            var c1:Array = [0x336600, 0x80ff00];
            var a1:Array = [255, 255];
            var r1:Array = [100, 255];
            var m1:Matrix = new Matrix();
            m1.createGradientBox(300, 200);
            g1.beginGradientFill(GradientType.LINEAR, c1, a1, r1, m1);
            g1.drawEllipse(0, 0, 300, 200);
```

```
            g1.endFill();
            addChild(backgroundShape);

            var foregroundShape:Shape = new Shape();
            var g2:Graphics = foregroundShape.graphics;
            var c2:Array = [0xff8000, 0x663300];
            var a2:Array = [255, 255];
            var r2:Array = [100, 255];
            var m2:Matrix = new Matrix();
            m2.createGradientBox(300, 200);
            g2.beginGradientFill(GradientType.LINEAR, c2, a2, r2, m2);
            g2.drawEllipse(100, 0, 300, 200);
            g2.endFill();
            addChild(foregroundShape);

            foregroundShape.blendShader = shader;
            foregroundShape.blendMode = BlendMode.SHADER;
        }
    }
}
```

Le code source du noyau du shader LumaLighten, utilisé pour la création du fichier « LumaLighten.pbj » de pseudo-code binaire Pixel Bender, est indiqué ci-dessous :

```
<languageVersion : 1.0;>
kernel LumaLighten
<
    namespace : "com.quasimondo.blendModes";
    vendor : "Quasimondo.com";
    version : 1;
    description : "Luminance based lighten blend mode";
>
{
    input image4 background;
    input image4 foreground;

    output pixel4 dst;

    const float3 LUMA = float3(0.212671, 0.715160, 0.072169);

    void evaluatePixel()
    {
        float4 a = sampleNearest(foreground, outCoord());
        float4 b = sampleNearest(background, outCoord());
        float luma_a = a.r * LUMA.r + a.g * LUMA.g + a.b * LUMA.b;
        float luma_b = b.r * LUMA.r + b.g * LUMA.g + b.b * LUMA.b;

        dst = luma_a > luma_b ? a : b;
    }
}
```

Pour plus d'informations sur l'utilisation des modes de fondu, voir « Application de modes de fondu » à la page 193.

Remarque : *lorsqu'un programme shader Pixel Bender est exécuté en tant que fusion dans Flash Player ou AIR, les fonctions d'échantillonnage et* `outCoord()` *ne se comportent pas comme dans les autres contextes. En mode de fusion, une fonction d'échantillonnage renvoie toujours le pixel en cours d'évaluation par le shader. Il est, par exemple, impossible d'ajouter un décalage à* `outCoord()` *pour échantillonner un pixel environnant. De même, si vous utilisez la fonction* `outCoord()` *dans un contexte autre qu'une fonction d'échantillonnage, ses coordonnées renvoient toujours 0. Il est ainsi impossible de se baser sur la position d'un pixel pour affecter la combinaison des images fusionnées.*

Utilisation d'un shader comme filtre

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Utiliser un shader comme filtre s'apparente à l'utilisation de tout autre filtre dans ActionScript. Lorsque vous utilisez un shader en tant que filtre, l'image filtrée (un objet d'affichage ou un objet BitmapData) est transmise au shader. Le shader utilise l'image d'entrée pour créer le résultat du filtre, qui correspond généralement à une version modifiée de l'image d'origine. Si l'objet filtré est un objet d'affichage, le résultat du filtre remplace l'objet d'affichage filtré à l'écran. Si l'objet filtré est un objet BitmapData, le résultat du shader devient le contenu de l'objet BitmapData dont la méthode `applyFilter()` est appelée.

Pour utiliser un shader en tant que filtre, vous commencez par créer l'objet Shader, comme indiqué à la section « Chargement ou intégration d'un shader » à la page 312. Vous créez ensuite un objet ShaderFilter lié à l'objet Shader. L'objet ShaderFilter est le filtre appliqué à l'objet filtré. Vous l'appliquez à un objet à l'instar de n'importe quel filtre. Vous le transmettez à la propriété `filters` d'un objet d'affichage ou vous appelez la méthode `applyFilter()` sur un objet BitmapData. Par exemple, le code suivant crée un objet ShaderFilter et applique le filtre à un objet d'affichage appelé `homeButton`.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

Lorsque vous utilisez un shader en tant que filtre, une entrée au moins doit lui être associée. Comme indiqué dans l'exemple, vous ne définissez pas la valeur d'entrée dans le code. L'objet d'affichage filtré ou l'objet BitmapData est défini en tant qu'image d'entrée. Si vous utilisez un shader qui attend plus d'une entrée, vous en spécifiez la valeur à partir de la deuxième entrée.

Dans certains cas, un filtre modifie les dimensions de l'image d'origine. Un effet d'ombre portée standard ajoute par exemple des pixels contenant l'ombre associée à l'image. Lorsque vous utilisez un shader qui modifie les dimensions de l'image, définissez les propriétés `leftExtension`, `rightExtension`, `topExtension` et `bottomExtension` pour indiquer l'écart approprié.

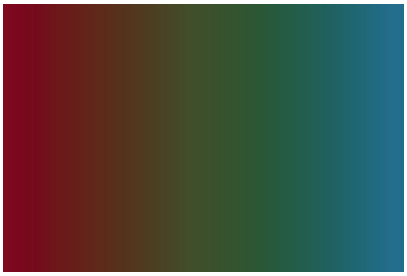
L'exemple suivant illustre l'utilisation d'un shader en tant que filtre. Dans cet exemple, le filtre inverse les valeurs des canaux rouge, vert et bleu d'une image. Il a pour résultat un « négatif » de l'image.

Remarque : *cet exemple utilise comme shader le noyau invertRGB.pbk Pixel Bender intégré à Pixel Bender Toolkit. Vous pouvez charger le code source du noyau à partir du répertoire d'installation de Pixel Bender Toolkit. Compilez le code source, puis enregistrez le fichier de pseudo-code binaire dans le même répertoire que le code source.*

Le code ActionScript important figure dans les deux méthodes suivantes :

- `init()` : la méthode `init()` est appelée lors du chargement de l'application. Dans cette méthode, le code charge le fichier de pseudo-code binaire du shader.

- `onLoadComplete()` : dans la méthode `onLoadComplete()`, le code crée l'objet Shader appelé `shader`. Il crée et dessine ensuite le contenu d'un objet appelé `target`. L'objet `target` est un rectangle rempli d'un dégradé linéaire rouge sur la gauche, jaune-vert au centre et bleu sur la droite. Si le filtre n'est pas appliqué, l'apparence de l'objet est la suivante :



Si le filtre est appliqué, les couleurs sont inversées, auquel cas l'apparence du rectangle est la suivante :

Ce code utilise à titre de shader le noyau « invertRGB.pbk » Pixel Bender d'exemple intégré à Pixel Bender Toolkit. Le code source figure dans le fichier « invertRGB.pbk », qui réside dans le répertoire d'installation de Pixel Bender Toolkit. Compilez le code source et enregistrez le fichier de pseudo-code binaire sous le nom « invertRGB.pbj » dans le même répertoire que votre code source ActionScript.

Vous trouverez ci-dessous le code ActionScript associé à cet exemple : Utilisez cette classe en tant que classe d'application principale d'un projet ActionScript uniquement dans Flash Builder, ou en tant que classe de document du fichier FLA dans Flash Professional :

```
package
{
    import flash.display.GradientType;
    import flash.display.Graphics;
    import flash.display.Shader;
    import flash.display.Shape;
    import flash.display.Sprite;
    import flash.filters.ShaderFilter;
    import flash.events.Event;
    import flash.geom.Matrix;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class InvertRGB extends Sprite
    {
        private var shader:Shader;
        private var loader:URLLoader;

        public function InvertRGB()
        {
            init();
        }

        private function init():void
        {
            loader = new URLLoader();
            loader.dataFormat = URLLoaderDataFormat.BINARY;
            loader.addEventListener(Event.COMPLETE, onLoadComplete);
            loader.load(new URLRequest("invertRGB.pbj"));
        }


        private function onLoadComplete(event:Event):void
        {
            shader = new Shader(loader.data);

            var target:Shape = new Shape();
            addChild(target);

            var g:Graphics = target.graphics;
            var c:Array = [0x990000, 0x445500, 0x007799];
            var a:Array = [255, 255, 255];
            var r:Array = [0, 127, 255];
            var m:Matrix = new Matrix();
            m.createGradientBox(w, h);
            g.beginGradientFill(GradientType.LINEAR, c, a, r, m);
            g.drawRect(10, 10, w, h);
            g.endFill();

            var invertFilter:ShaderFilter = new ShaderFilter(shader);
            target.filters = [invertFilter];
        }
    }
}
```

Pour plus d'informations sur l'application de filtres, voir « Création et application de filtres » à la page 277.

Utilisation d'un shader en mode autonome

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Lorsque vous utilisez un shader en mode autonome, son traitement s'exécute indépendamment de l'usage prévu du résultat. Vous spécifiez le shader à exécuter, définissez les valeurs d'entrée et de paramètres, puis stipulez un objet dans lequel seront placées les données résultantes. Vous pouvez utiliser un shader en mode autonome pour deux raisons :

- Traitement de données non graphiques : en mode autonome, vous pouvez choisir de transmettre des données binaires ou numériques arbitraires au shader au lieu de données d'image bitmap. Le cas échéant, le résultat du shader peut être renvoyé sous forme de données binaires ou numériques en plus des données d'image bitmap.
- Traitement en arrière-plan : lorsque vous exécutez un shader en mode autonome, il est exécuté en mode asynchrone par défaut. Cela signifie que le shader s'exécute en arrière-plan pendant que votre application continue à tourner et que votre code est averti une fois le traitement du shader terminé. Vous pouvez utiliser un shader sans pour autant bloquer l'interface utilisateur de l'application ou tout autre traitement, même si la durée de son exécution est élevée.

Un objet ShaderJob permet d'exécuter un shader en mode autonome. Vous commencez par créer un objet ShaderJob et vous le liez à l'objet Shader qui représente le shader à exécuter :

```
var job:ShaderJob = new ShaderJob(myShader);
```

Vous définissez ensuite toute valeur d'entrée ou de paramètre attendue par le shader. Si vous exécutez le shader en arrière-plan, vous enregistrez également un écouteur associé à l'événement `complete` de l'objet ShaderJob. Votre écouteur est appelé lorsque le shader termine sa tâche :

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

Vous créez ensuite un objet dans lequel est écrit le résultat de l'opération du shader, une fois celle-ci terminée. Vous affectez cet objet à la propriété `target` de l'objet ShaderJob :

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

Affectez une occurrence de BitmapData à la propriété `target` si vous utilisez l'objet ShaderJob pour exécuter le traitement de l'image. Si vous traitez des données binaires ou numériques, affectez un objet ByteArray ou une occurrence de Vector.<Number> à la propriété `target`. Si tel est le cas, vous devez définir les propriétés `width` et `height` de l'objet ShaderJob pour stipuler le volume de données à créer dans l'objet `target`.

Remarque : *vous pouvez définir les propriétés `shader`, `target`, `width` et `height` de l'objet ShaderJob en une seule étape. Pour ce faire, transmettez les arguments au constructeur `ShaderJob()`, comme suit : `var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);`.*

Lorsque vous êtes prêt à exécuter le shader, vous appelez la méthode `start()` de l'objet ShaderJob :

```
job.start();
```

Par défaut, appeler `start()` entraîne l'exécution asynchrone de l'objet ShaderJob. Dans ce cas, l'exécution du programme continue et passe immédiatement à la ligne suivante de code au lieu d'attendre que le shader soit terminé. Une fois l'opération du shader terminée, l'objet ShaderJob appelle ses écouteurs d'événement `complete` et les avertit. A ce stade (en d'autres termes, dans le corps de votre écouteur d'événement `complete`), l'objet `target` contient le résultat de l'opération du shader.

Remarque : *au lieu d'utiliser l'objet propriété* `target`*, vous pouvez extraire directement le résultat du shader de l'objet événement qui est transmis à la méthode d'écouteur. L'objet événement est une occurrence de ShaderEvent. L'objet ShaderEvent possède trois propriétés susceptibles d'être utilisées pour accéder au résultat, selon le type de données de l'objet défini en tant que propriété* `target` *:* `ShaderEvent.bitmapData`*,* `ShaderEvent.byteArray` *et* `ShaderEvent.vector`*.*

Vous pouvez également transmettre un argument `true` à la méthode `start()`. Dans ce cas, l'opération du shader s'exécute en mode synchrone. Tout le code (y compris l'interaction avec l'interface utilisateur et tout autre événement) est interrompu pendant l'exécution du shader. Une fois le shader terminé, l'objet `target` en contient le résultat et le programme passe à la ligne de code suivante.

```
job.start(true);
```

Chapitre 16 : Utilisation des clips

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe MovieClip est la classe de base des animations et des symboles de clip créés dans l'environnement de développement Adobe® Flash®. Elle possède tous les comportements et toutes les fonctionnalités des objets d'affichage, mais intègre des propriétés et méthodes supplémentaires qui permettent de contrôler son scénario.

Principes de base des clips

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les clips sont un élément capital pour les créateurs de contenu animé dans l'environnement de création Flash et pour contrôler ce contenu avec ActionScript. Lorsque vous créez un symbole de clip dans Flash, le programme ajoute ce symbole à la bibliothèque du document Flash. Par défaut, ce symbole devient une occurrence de la classe MovieClip et, de ce fait, possède les propriétés et méthodes de cette classe.

Lorsqu'une occurrence d'un symbole de clip est placée sur la scène, la progression du scénario de ce clip s'effectue automatiquement (si le clip comporte plusieurs images), sauf si cette lecture est modifiée en ActionScript. Le scénario est l'élément distinctif de la classe MovieClip. Il permet de créer des animations par interpolation de mouvement ou de forme via l'interface de Flash. Par contre, un objet d'affichage issu de la classe Sprite peut uniquement être animé par programmation, en modifiant ses valeurs.

Dans les versions précédentes d'ActionScript, la classe MovieClip constituait la classe de base de toutes les occurrences de la scène. En ActionScript 3.0, un clip est désormais un objet d'affichage parmi d'autres, tous susceptibles de s'afficher à l'écran. S'il n'est pas nécessaire de définir un scénario pour la fonction d'un objet d'affichage, l'utilisation de la classe Shape au lieu de la classe MovieClip peut accroître les performances d'affichage. Pour plus d'informations sur le choix des objets d'affichage en fonction de la tâche prévue, voir « Sélection d'une sous-classe de DisplayObject » à la page 179.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs aux clips :

Fichier SWF AVM1 Fichier SWF créé en ActionScript 1.0 ou ActionScript 2.0, et généralement destiné aux versions 8 ou antérieures de Flash Player.

Fichier SWF AVM2 Fichier SWF créé en ActionScript 3.0 pour Adobe Flash Player 9 ou ultérieur ou Adobe AIR.

Fichier SWF externe Fichier SWF créé séparément du fichier SWF du projet, et destiné à être chargé et lu dans celui-ci.

Image Elément de base du scénario. Comme les images de films, chaque image est une « photographie » et c'est la lecture rapide des images en séquence qui produit l'effet d'animation.

Scénario Métaphore représentant la série d'images qui composent la séquence d'animation d'un clip. Le scénario d'un objet MovieClip est l'équivalent du scénario de cet objet dans l'environnement de création Flash.

Tête de lecture Marqueur identifiant la position (l'image) dans le scénario qui est affichée à un moment donné.

Utilisation des objets MovieClip

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous publiez un fichier SWF, Flash convertit toutes les occurrences de symboles de clips sur la scène en objets MovieClip. Pour qu'un symbole de clip soit disponible en ActionScript, donnez-lui un nom d'occurrence dans le champ Nom de l'occurrence de l'Inspecteur des Propriétés. Lors de la création du fichier SWF, Flash génère le code qui crée l'occurrence de MovieClip sur la scène et déclare une variable avec ce nom d'occurrence. Si les clips que vous avez nommés sont imbriqués dans d'autres clips nommés, ces clips enfant sont traités comme des propriétés du clip parent : vous pouvez accéder à tous les clips de cette hiérarchie en utilisant la syntaxe à point. Par exemple, si une occurrence de clip appelée `childClip` est imbriquée dans une autre appelée `parentClip`, vous pouvez lancer sa lecture en appelant le code suivant :

```
parentClip.childClip.play();
```

Remarque : *les occurrences enfant placées sur la scène dans l'outil de programmation de Flash ne sont pas accessibles par le code depuis l'intérieur du constructeur d'une occurrence parent car, à ce stade de l'exécution du code, elles n'ont pas été créées. Avant d'accéder à l'enfant, le parent doit soit créer l'occurrence enfant par du code, soit retarder l'accès à une fonction de rappel qui écoute l'enfant en vue de la distribution de son événement* `Event.ADDED_TO_STAGE`.

Si de nombreuses méthodes et propriétés de la classe MovieClip d'ActionScript 2.0 restent les mêmes, d'autres ont changé. Toutes les propriétés qui étaient préfixées d'un trait de soulignement ont été renommées. Par exemple, les propriétés `_width` et `_height` sont désormais accessibles sous le nom `width` et `height`, `_xscale` et `_yscale` sous le nom `scaleX` et `scaleY`. Pour obtenir la liste complète des propriétés et méthodes de la classe MovieClip, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Contrôle de la lecture d'un clip

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash utilise la métaphore du scénario pour traduire une animation ou un changement d'état. Tout élément visuel qui a recours à un scénario est soit un objet MovieClip, soit une extension de la classe MovieClip. Bien qu'il soit possible en ActionScript de commander l'arrêt ou la lecture d'un clip et le passage à un autre point du scénario, il n'est pas possible de créer dynamiquement un scénario ni d'ajouter du contenu dans une image spécifique. Seul l'outil de programmation Flash le permet.

Pendant la lecture, le clip progresse le long du scénario à une vitesse fixée par la cadence d'image du fichier SWF. Vous pouvez également ignorer ce paramètre et le remplacer par la propriété `Stage.frameRate` dans ActionScript.

Lecture et arrêt des clips

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes `play()` et `stop()` permettent d'effectuer des manipulations simples sur un clip tout au long du scénario. Par exemple, supposons qu'un symbole de clip sur la scène contienne une animation représentant une bicyclette qui traverse l'écran, et dont le nom d'occurrence est `bicycle`. Si le code suivant est associé à une image-clé du scénario principal,

```
bicycle.stop();
```

la bicyclette ne se déplace pas (son animation n'est pas mise en lecture). Le mouvement de la bicyclette peut être déclenché par une action de l'utilisateur. Par exemple, avec un bouton nommé `startButton`, le code suivant (dans une image-clé du scénario principal) déclenche l'animation si l'utilisateur clique sur ce bouton :

```
// This function will be called when the button is clicked. It causes the
// bicycle animation to play.
function playAnimation(event:MouseEvent):void
{
    bicycle.play();
}
// Register the function as a listener with the button.
startButton.addEventListener(MouseEvent.CLICK, playAnimation);
```

Avance rapide et rembobinage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes `play()` et `stop()` ne sont pas les seules méthodes commandant l'animation d'un clip. Vous pouvez également avancer et reculer manuellement la tête de lecture à l'aide des méthodes `nextFrame()` et `prevFrame()`. L'appel de l'une de ces deux méthodes arrête la lecture et fait avancer le clip ou le rembobine d'une image.

L'utilisation de la méthode `play()` revient à appeler `nextFrame()` chaque fois qu'un événement `enterFrame` est déclenché pour cet objet clip. Vous pouvez donc envisager de lire le clip `bicycle` en arrière en ajoutant un écouteur pour l'événement `enterFrame` et en indiquant à `bicycle`, dans la fonction écouteur, de reculer d'une image, comme suit :

```
// This function is called when the enterFrame event is triggered, meaning
// it's called once per frame.
function everyFrame(event:Event):void
{
    if (bicycle.currentFrame == 1)
    {
        bicycle.gotoAndStop(bicycle.totalFrames);
    }
    else
    {
        bicycle.prevFrame();
    }
}
bicycle.addEventListener(Event.ENTER_FRAME, everyFrame);
```

En lecture normale, si un clip contient plusieurs images, il tourne en boucle lors de la lecture, c'est-à-dire qu'il revient à l'image 1 une fois qu'il a passé la dernière image. Si vous utilisez `prevFrame()` ou `nextFrame()`, ce comportement ne se produit pas automatiquement (l'appel de `prevFrame()` lorsque la tête de lecture est sur l'image 1 ne déplace pas la tête de lecture à la dernière image). Dans l'exemple ci-dessus, la condition `if` vérifie si le clip est revenu à la première image et fait alors passer le clip à la dernière image, créant ainsi une boucle continue de lecture en arrière.

Déplacement vers une autre image et utilisation des étiquettes d'image

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le déplacement d'un clip à une nouvelle image est une opération simple. Il suffit d'appeler `gotoAndPlay()` ou `gotoAndStop()` en spécifiant le numéro de l'image cible comme paramètre. Vous pouvez également transmettre une chaîne correspondant au nom de l'étiquette d'image. Il est possible d'attribuer une étiquette à toute image du scénario. Pour ce faire, sélectionnez une image du scénario, puis tapez un nom dans le champ Etiquette d'image de l'Inspecteur des Propriétés.

L'utilisation des étiquettes d'image plutôt que des numéros présente des avantages évidents pour créer un clip complexe. Si le nombre d'images, de calques et d'interpolations d'une animation est élevé, envisagez d'étiqueter les images importantes de manière évocatrice, en faisant référence aux changements de comportement dans le clip (par exemple « départ », « marche » ou « course »). Cette technique permet d'améliorer la lisibilité du code et offre plus de souplesse, puisque les appels ActionScript destinés à une image étiquetée pointent sur une référence uniquement (l'étiquette) plutôt que sur une image numérotée. Si vous décidez par la suite de déplacer un segment de l'animation vers une autre image, il ne sera pas nécessaire de modifier le code ActionScript si vous conservez les mêmes étiquettes pour toutes les images au nouvel emplacement.

Pour représenter des étiquettes en code, ActionScript 3.0 comporte la classe FrameLabel. Chaque occurrence de cette classe représente une étiquette d'image unique, et possède une propriété `name` qui représente le nom de l'étiquette tel qu'il a été indiqué dans l'Inspecteur des Propriétés, ainsi qu'une propriété `frame` qui représente le numéro de l'image pour laquelle l'étiquette est placée dans le scénario.

Pour permettre d'accéder aux occurrences de FrameLabel associées à une occurrence de clip, la classe MovieClip comporte deux propriétés qui renvoient directement des objets FrameLabel. La propriété `currentLabels` renvoie un tableau composé de tous les objets FrameLabel présents sur l'ensemble du scénario d'un clip. La propriété `currentLabel` renvoie une chaîne contenant le nom de l'étiquette d'image trouvée récemment sur le scénario.

Supposons que vous ayez créé un clip nommé `Robot` et que vous ayez étiqueté les différents états de l'animation. Vous pouvez définir une condition pour vérifier la propriété `currentLabel` afin d'accéder à l'état actuel de `Robot`, comme dans le code suivant :

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

Flash Player 11.3 et AIR 3.3 ont ajouté l'événement `frameLabel` à la classe FrameLabel. Vous pouvez affecter un gestionnaire d'événement à l'occurrence de FrameLabel qui représente l'étiquette d'une trame. L'événement est distribué lorsque la tête de lecture accède à la trame.

L'exemple suivant crée une occurrence de FrameLabel pour la seconde étiquette de trame dans l'objet Array des étiquettes de trames du MovieClip. Il enregistre ensuite un gestionnaire d'événement pour l'événement `frameLabel` :

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

Utilisation des séquences

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans l'environnement de création Flash, vous pouvez utiliser les séquences pour démarquer une série de scénarios qu'un fichier SWF doit suivre. Grâce au second paramètre de la méthode `gotoAndPlay()` ou `gotoAndStop()`, vous pouvez spécifier la séquence sur laquelle la tête de lecture doit se placer. Tous les fichiers FLA commencent par une séquence, à laquelle vous pouvez ajouter le nombre de séquences de votre choix.

L'utilisation des séquences n'est pas toujours conseillée, car elle présente un certain nombre d'inconvénients. La maintenance d'un document Flash qui contient de nombreuses séquences peut être difficile, en particulier dans les environnements où plusieurs auteurs interviennent. Un nombre élevé de séquences peut également s'avérer inefficace de point de vue de la bande passante, car le processus de publication implique la fusion de toutes les séquences en un seul scénario. L'ensemble des séquences est alors téléchargé en mode progressif, même si elles ne sont jamais lues. C'est pourquoi l'utilisation de plusieurs séquences est largement déconseillée, à moins qu'elle soit nécessaire à l'organisation de plusieurs animations basées sur des scénarios.

La propriété `scenes` de la classe MovieClip renvoie un tableau des objets Scene représentant toutes les séquences du fichier SWF. La propriété `currentScene` renvoie un objet Scene représentant la séquence qui est en cours de lecture.

La classe Scene a des propriétés qui contiennent des informations sur la séquence. La propriété `labels` renvoie un tableau des objets FrameLabel utilisés au sein de la séquence. La propriété `name` renvoie le nom de la séquence sous forme de chaîne. La propriété `numFrames` renvoie un entier représentant le nombre total d'images dans la séquence.

Création d'objets MovieClip à l'aide d'ActionScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'ajout de contenu s'effectue dans Flash en faisant glisser des éléments de la bibliothèque sur la scène. Mais il ne s'agit pas là de la seule méthode. Pour des projets plus complexes, les développeurs expérimentés préfèrent souvent créer les clips par programmation. Cette approche présente plusieurs avantages : réutilisation facile du code, vitesse de compilation accrue et disponibilité de modifications plus sophistiquées réservées à ActionScript.

La nouvelle API de liste d'affichage d'ActionScript 3.0 simplifie le processus de création dynamique d'objets MovieClip. La possibilité d'instancier directement une occurrence de MovieClip, séparément de son ajout à la liste d'affichage, offre beaucoup de souplesse et de simplicité sans sacrifier tout contrôle.

En ActionScript 3.0, lorsqu'une occurrence de clip (ou de tout autre objet d'affichage) est créée par code, elle est invisible tant qu'elle n'a pas été ajoutée à la liste d'affichage à l'aide de la méthode `addChild()` ou `addChildAt()` d'un conteneur d'objets d'affichage Vous pouvez ainsi créer un clip et définir ses propriétés, et même appeler ses méthodes, avant qu'il apparaisse à l'écran. Pour plus d'informations sur l'utilisation de la liste d'affichage, voir « Utilisation de conteneurs d'objets d'affichage » à la page 165.

Exportation des symboles de bibliothèque pour ActionScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, il est impossible de créer dynamiquement des occurrences de symboles de clips dans la bibliothèque d'un document Flash (c'est-à-dire de les créer entièrement en ActionScript). En effet, l'exportation de chaque symbole pour une utilisation en ActionScript accroît la taille du fichier SWF. Or, les symboles ne sont pas tous destinés à une utilisation sur la scène. C'est pourquoi, pour qu'un symbole soit disponible en ActionScript, vous devez indiquer spécifiquement que ce symbole doit être exporté pour ActionScript.

Pour exporter un symbole pour ActionScript :

1. Sélectionnez le symbole dans le panneau Bibliothèque et ouvrez sa boîte de dialogue Propriétés.
2. Si nécessaire, activez les paramètres avancés.
3. Dans la boîte de dialogue Liaison, activez l'option Exporter pour ActionScript.

   Les champs Classe et Classe de base sont alors activés.

   Par défaut, le champ Classe contient le nom du symbole sans espaces (par exemple, un symbole nommé « Maison Rouge » devient « MaisonRouge »). Pour spécifier que le symbole doit utiliser une classe personnalisée pour son comportement, saisissez le nom complet de cette classe, package compris. Si vous voulez avoir la possibilité de créer des occurrences du symbole en ActionScript, sans avoir pour autant besoin de comportements supplémentaires, vous pouvez laisser ce nom de classe tel quel.

   Par défaut, le champ Classe de base a la valeur `flash.display.MovieClip`. Si vous voulez que votre symbole étende les fonctionnalités d'une autre classe personnalisée, vous pouvez remplacer cette valeur par le nom de votre classe, sous réserve que celle-ci étende la classe Sprite (ou MovieClip).
4. Cliquez sur OK pour enregistrer les changements.

   A ce stade, si Flash ne détecte pas de fichier SWC lié ou de fichier ActionScript externe contenant une définition de la classe spécifiée (par exemple, si vous n'aviez pas besoin d'ajouter un comportement supplémentaire pour le symbole), un message d'avertissement apparaît :

   Le chemin de classe ne contient pas de définition pour cette classe. Une définition sera générée automatiquement dans le fichier SWF lors de l'exportation.

   Vous pouvez ignorer cet avertissement si le symbole de bibliothèque ne nécessite pas de fonctionnalités en dehors de celles de la classe MovieClip.

Si vous n'indiquez aucune classe pour votre symbole, Flash crée pour lui une classe du type de celle-ci :

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

Si vous ne souhaitez pas ajouter des fonctionnalités ActionScript au symbole, ajoutez les propriétés et méthodes nécessaires à la structure de code suivante. Supposons par exemple que vous disposez d'un symbole de clip contenant un cercle de 50 pixels de diamètre, dont le paramètre de liaison est Exporter pour ActionScript et dont la classe est Circle. Le code suivant, lorsqu'il est placé dans un fichier Circle.as, étend la classe MovieClip et fournit au symbole les méthodes supplémentaires `getArea()` et `getCircumference()` :

```
package
{
    import flash.display.MovieClip;

    public class Circle extends MovieClip
    {
        public function Circle()
        {
        }

        public function getArea():Number
        {
        // The formula is Pi times the radius squared.
        return Math.PI * Math.pow((width / 2), 2);
        }

        public function getCircumference():Number
        {
        // The formula is Pi times the diameter.
        return Math.PI * width;
        }
    }
}
```

Le code suivant, placé dans une image-clé à l'image 1 du document Flash, crée une occurrence du symbole et l'affiche à l'écran :

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

Ce code montre comment utiliser l'instanciation ActionScript au lieu de faire glisser des actifs individuels vers la scène. Il crée un cercle qui présente toutes les propriétés d'un clip et possède les méthodes personnalisées définies dans votre classe Circle. Il s'agit d'un exemple tout à fait élémentaire ; un symbole de bibliothèque peut spécifier un nombre illimité de propriétés et de méthodes dans sa classe.

L'instanciation en ActionScript est un processus puissant, car il permet de créer dynamiquement de nombreuses occurrences qu'il serait particulièrement laborieux de créer manuellement. Elle vous apporte en outre une grande souplesse, puisque vous pouvez personnaliser les propriétés de chaque occurrence dès sa création. Pour saisir l'importance de ces avantages, vous pouvez utiliser une boucle pour créer dynamiquement plusieurs occurrences de Circle. Après avoir ajouté le symbole Circle et la classe correspondante, décrits précédemment, dans la bibliothèque de votre document Flash, placez le code suivant dans une image-clé à l'image 1 :

```
import flash.geom.ColorTransform;

var totalCircles:uint = 10;
var i:uint;
for (i = 0; i < totalCircles; i++)
{
    // Create a new Circle instance.
    var c:Circle = new Circle();
    // Place the new Circle at an x coordinate that will space the circles
    // evenly across the Stage.
    c.x = (stage.stageWidth / totalCircles) * i;
    // Place the Circle instance at the vertical center of the Stage.
    c.y = stage.stageHeight / 2;
    // Change the Circle instance to a random color
    c.transform.colorTransform = getRandomColor();
    // Add the Circle instance to the current timeline.
    addChild(c);
}

function getRandomColor():ColorTransform
{
    // Generate random values for the red, green, and blue color channels.
    var red:Number = (Math.random() * 512) - 255;
    var green:Number = (Math.random() * 512) - 255;
    var blue:Number = (Math.random() * 512) - 255;

    // Create and return a ColorTransform object with the random colors.
    return new ColorTransform(1, 1, 1, 1, red, green, blue, 0);
}
```

Cet exemple illustre la rapidité avec laquelle vous pouvez créer et personnaliser plusieurs occurrences d'un symbole à l'aide de code. Chaque occurrence est positionnée en fonction du compteur de boucle. Ensuite une couleur lui est attribuée au hasard à l'aide de la propriété `transform` (dont Circle hérite en étendant la classe MovieClip).

Chargement d'un fichier SWF externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En ActionScript 3.0, les fichiers SWF sont chargés avec la classe Loader. Pour charger un fichier SWF externe, vous devez définir quatre étapes en ActionScript :

1. Créer un objet URLRequest avec l'adresse URL du fichier.
2. Créer un objet Loader.
3. Appeler la méthode `load()` de l'objet Loader en lui passant en paramètre l'occurrence de l'objet URLRequest.
4. Appeler la méthode `addChild()` pour un conteneur d'objet d'affichage (par exemple le scénario principal d'un document Flash) afin d'ajouter l'occurrence de Loader à la liste d'affichage

Le code final se présente ainsi :

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

Le même code peut être utilisé pour charger un fichier image externe (image JPEG, GIF ou PNG) en spécifiant l'adresse URL de ce fichier à la place de celle d'un fichier SWF. Contrairement à un fichier image, un fichier SWF peut contenir du code ActionScript. Ainsi, bien que le processus de chargement d'un fichier SWF soit identique à celui d'une image, le fichier SWF à l'origine du chargement et le fichier SWF chargé doivent se trouver dans le même sandbox de sécurité si vous souhaitez utiliser ActionScript pour communiquer avec le fichier SWF externe. En outre, si ce fichier externe contient des classes qui partagent le même espace de noms que les classes du fichier SWF qui le charge, il peut s'avérer nécessaire de créer un domaine d'application pour le fichier chargé afin d'éviter d'éventuels conflits d'espace de nom. Pour plus d'informations sur la sécurité et le domaine d'application, voir « Utilisation de domaines d'application » à la page 152 et « Chargement de contenu » à la page 1104.

Une fois que le fichier SWF externe est chargé, il devient accessible via la propriété `Loader.content`. Si ce fichier est publié en ActionScript 3.0, il s'agira soit d'un clip, soit d'un sprite, selon la classe qu'il étend.

Il existe quelques différences entre le chargement d'un fichier SWF dans Adobe AIR pour iOS et le chargement dans d'autres plates-formes. Pour plus d'informations, voir « Chargement de fichiers SWF dans AIR pour iOS » à la page 209.

Considérations relatives au chargement de fichiers SWF de version ancienne

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si un fichier SWF externe a été publié dans une version antérieure d'ActionScript, il faut tenir compte de restrictions importantes. Contrairement à un fichier SWF ActionScript 3.0 qui s'exécute dans AVM2 (ActionScript Virtual Machine 2), un fichier SWF publié en ActionScript 1.0 ou 2.0 s'exécute dans AVM1 (ActionScript Virtual Machine 1).

Le chargement d'un fichier SWF ActionScript 1.0 ou 2.0 dans un fichier SWF ActionScript 3.0 présente d'importantes différences par rapport au chargement d'un fichier SWF ActionScript 3.0. Flash Player garantit une compatibilité totale avec les versions précédentes et les contenus publiés antérieurement. Tout contenu qui s'exécute dans les versions antérieures de Flash Player s'exécute dans les versions de Flash Player qui gèrent ActionScript 3.0. Tenez cependant compte des restrictions suivantes :

- Le code ActionScript 3.0 peut charger un fichier SWF écrit en ActionScript 1.0 ou 2.0. Lorsque le chargement d'un fichier SWF ActionScript 1.0 ou 2.0 aboutit, l'objet chargé (la propriété `Loader.content`) est un objet AVM1Movie. Une occurrence d'AVM1Movie n'est pas identique à une occurrence de MovieClip. C'est un objet d'affichage qui, contrairement à un clip, ne comporte pas de méthodes ni de propriétés liées au scénario. Le fichier SWF AVM2 parent ne peut pas accéder aux propriétés, méthodes ou objets de l'objet AVM1Movie chargé.
- Les fichiers SWF écrits en ActionScript 1.0 ou 2.0 ne peuvent pas charger les fichiers SWF écrits en ActionScript 3.0. Cela signifie que les fichiers SWF créés dans Flash 8, Flex Builder 1.5 ou une version antérieure ne peuvent pas charger les fichiers SWF écrits en ActionScript 3.0.

  Il existe une seule exception à cette règle : un fichier SWF écrit en ActionScript 2.0 peut être remplacé par un fichier SWF écrit en ActionScript 3.0, sous réserve que le fichier écrit en ActionScript 2.0 n'ait effectué aucun chargement à quelque niveau que ce soit. Pour ce faire, le fichier SWF écrit en ActionScript 2.0 doit appeler `loadMovieNum()` et transmettre la valeur 0 au paramètre `level`.
- En règle générale, les fichiers SWF écrits en ActionScript 1.0 ou 2.0 doivent faire l'objet d'une migration pour fonctionner conjointement avec des fichiers SWF écrits en ActionScript 3.0. Supposons, par exemple, que vous ayez créé un lecteur multimédia avec ActionScript 2.0. Ce lecteur multimédia charge des contenus divers également créés en ActionScript 2.0. Il est impossible de créer un contenu en ActionScript 3.0 et de le charger dans le lecteur multimédia. Vous devez effectuer une migration du lecteur vers ActionScript 3.0.

  Néanmoins, si vous créez un lecteur multimédia en ActionScript 3.0, il peut effectuer de simples chargements de votre contenu en ActionScript 2.0.

Le tableau ci-après récapitule les limites des versions précédentes de Flash Player relatives au chargement de contenus plus récents et à l'exécution de code, ainsi que les restrictions liées à la programmation croisée entre les fichiers SWF écrits en différentes versions d'ActionScript.

Fonctionnalité prise en charge	**Flash Player 7**	**Flash Player 8**	**Flash Player 9 et 10**
Peut charger les fichiers SWF pour	version 7 et antérieure	version 8 et antérieure	version 9 (ou 10) et antérieure
Machine virtuelle incluse	AVM1	AVM1	AVM1 et AVM2
Exécute les fichiers SWF écrits en ActionScript	1.0 et 2.0	1.0 et 2.0	1.0, 2.0 et 3.0

Dans le tableau ci-dessous, « Fonctionnalité prise en charge » fait référence au contenu lisible dans Flash Player 9 ou une version ultérieure. Le contenu lisible dans Flash Player version 8 ou antérieure peut être chargé, affiché, exécuté et programmé avec ActionScript 1.0 et 2.0 uniquement.

Fonctionnalité prise en charge	**Contenu créé en ActionScript 1.0 et 2.0**	**Contenu créé en ActionScript 3.0**
Peut charger du contenu et exécuter le code de contenus créé dans	ActionScript 1.0 et 2.0 uniquement	ActionScript 1.0 et 2.0 et ActionScript 3.0
Peut programmer le contenu créé dans	ActionScript 1.0 et 2.0 uniquement (ActionScript 3.0 via LocalConnection)	ActionScript 1.0 et 2.0 via LocalConnection ActionScript 3.0

Exemple de clip : RuntimeAssetsExplorer

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La fonctionnalité Exporter pour ActionScript présente des avantages particuliers lorsque des bibliothèques peuvent être réutilisées sur plusieurs projets. Si Flash Player ou AIR exécute un fichier SWF, les symboles ayant été exportés dans ActionScript sont disponibles pour tous les fichiers SWF au sein du même sandbox de sécurité que le fichier SWF qui le charge. De cette manière, un seul document Flash peut générer un fichier SWF destiné uniquement à accueillir des éléments graphiques. Cette technique est très utile pour les grands projets dans lesquels les concepteurs graphiques chargés des éléments visuels peuvent travailler en parallèle avec les développeurs qui créent une « enveloppe » SWF qui chargera les fichiers SWF des éléments graphiques au moment de l'exécution. Cette méthode peut vous servir dans la maintenance d'un ensemble de versions de fichiers dans lesquelles les actifs graphiques ne sont pas dépendants du développement de la programmation.

L'application RuntimeAssetsExplorer charge tout fichier SWF qui est une sous-classe de RuntimeAsset et permet de consulter les éléments disponibles de ce fichier SWF. Cet exemple illustre les points suivants :

- Chargement d'un fichier SWF externe à l'aide de `Loader.load()`
- Création dynamique d'un symbole de bibliothèque exporté pour ActionScript
- Contrôle de la lecture du MovieClip avec ActionScript

Avant de commencer, notez que tous les fichiers SWF devant s'exécuter dans Flash Player doivent se trouver dans le même sandbox de sécurité. Pour plus d'informations, voir « Sandbox de sécurité » à la page 1087.

Pour obtenir les fichiers d'application associés à cet exemple, téléchargez les exemples Flash Professional. Les fichiers de l'application RuntimeAssetsExplorer se trouvent dans le dossier Samples/Chapters/RuntimeAssetsExplorer. L'application se compose des fichiers suivants :

Fichier	Description
RuntimeAssetsExample.mxml ou RuntimeAssetsExample.fla	Interface utilisateur de l'application pour Flex (MXML) ou Flash (FLA).
RuntimeAssetsExample.as	Classe document pour l'application Flash (FLA).
GeometricAssets.as	Classe exemple qui implémente l'interface RuntimeAsset.
GeometricAssets.fla	Fichier FLA lié à la classe GeometricAssets (classe de document du fichier FLA) contenant les symboles exportés pour ActionScript.
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	Interface qui définit les méthodes attendues par tous les fichiers SWF d'actifs d'exécution qui seront chargés dans l'explorateur.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	Classe du symbole de bibliothèque dont la forme est un rectangle pivotant.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	Classe du symbole de bibliothèque dont la forme est une étoile pivotante.

Etablissement de l'interface de bibliothèque d'exécution

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour que l'explorateur interagisse convenablement avec une bibliothèque SWF, la structure des bibliothèques d'éléments doit être formalisée. Pour ce faire, nous allons créer une interface, que l'on pourrait comparer à une classe dans la mesure où elle représente un modèle de définition des méthodes qui définissent une structure attendue. Par contre, à l'inverse d'une classe, elle ne comporte pas les corps des méthodes. L'interface est un moyen de communication pour la bibliothèque d'éléments et l'explorateur. Chaque fichier SWF d'éléments chargé dans le navigateur implémentera cette interface. Pour plus d'informations sur l'utilité des interfaces, voir Interfaces dans *Formation à ActionScript 3.0*.

L'interface RuntimeLibrary est très simple, puisque nous avons uniquement besoin d'une fonction qui fournisse à l'explorateur un tableau de chemins de classe pour les symboles à exporter disponibles dans la bibliothèque. Cette interface utilise pour ce faire une seule méthode : `getAssets()`.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

Création de fichiers SWF de bibliothèque d'actifs

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La définition de l'interface RuntimeLibrary permet de créer plusieurs fichiers SWF de bibliothèque d'actifs, qui pourront être chargés dans un autre fichier SWF. L'élaboration d'une bibliothèque SWF d'actifs repose sur quatre tâches :

- Création d'une classe pour le fichier SWF de bibliothèque d'actifs
- Création de classes pour les différents éléments contenus dans la bibliothèque
- Création des éléments graphiques eux-mêmes
- Association des éléments graphiques à des classes et publication du fichier SWF de bibliothèque

Création d'une classe pour implémenter l'interface RuntimeLibrary

Nous allons ensuite créer la classe GeometricAssets qui implémente l'interface RuntimeLibrary, et sera la classe du document FLA. Le code de cette classe est très proche de celui de l'interface RuntimeLibrary, la différence étant que la définition de classe comporte le corps de la méthode `getAssets()`.

```
package
{
    import flash.display.Sprite;
    import com.example.programmingas3.runtimeassetexplorer.RuntimeLibrary;

    public class GeometricAssets extends Sprite implements RuntimeLibrary
    {
        public function GeometricAssets() {

        }
        public function getAssets():Array {
            return [ "com.example.programmingas3.runtimeassetexplorer.AnimatingBox",
                     "com.example.programmingas3.runtimeassetexplorer.AnimatingStar" ];
        }
    }
}
```

Si nous devions créer une deuxième bibliothèque d'exécution, nous pourrions créer un autre fichier FLA reposant sur une autre classe (AnimationAssets, par exemple) qui assure sa propre implémentation de `getAssets()`.

Création de classes pour chaque élément MovieClip

Pour les besoins de cet exemple, nous étendrons simplement la classe MovieClip, sans ajouter de fonctionnalité aux éléments. Le code suivant destiné à AnimatingStar est semblable à celui de AnimatingBox :

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

Publication de la bibliothèque

Nous allons maintenant relier les actifs de classe MovieClip à la nouvelle classe en créant un fichier FLA et en entrant GeometricAssets dans le champ Classe du document de l'Inspecteur des propriétés. Pour les besoins de cet exemple, nous allons créer deux formes très simples qui utilisent une interpolation de scénario pour effectuer une rotation horaire sur 360 images. Les deux symboles `animatingBox` et `animatingStar` sont définis pour l'exportation vers ActionScript et leurs champs Classe correspondent aux chemins de classe respectifs spécifiés dans l'implémentation de `getAssets()`. La classe de base par défaut `flash.display.MovieClip` est conservée, puisque nous voulons créer des sous-classes pour les méthodes MovieClip standard.

Après définition des paramètres d'exportation de votre symbole, publiez le fichier FLA. Vous disposez maintenant de votre première bibliothèque d'exécution. Ce fichier SWF pourrait être chargé dans un autre fichier SWF AVM2, dans lequel les symboles AnimatingBox et AnimatingStar seraient disponibles.

Chargement de la bibliothèque dans un autre fichier SWF

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le dernier élément fonctionnel à traiter est l'interface utilisateur de l'explorateur. Dans cet exemple, le chemin d'accès à la bibliothèque d'exécution est codé en dur dans la variable `ASSETS_PATH`. Vous pourriez également utiliser la classe FileReference, par exemple pour créer une interface qui recherche un fichier SWF particulier sur le disque dur.

Une fois que la bibliothèque d'exécution est chargée, Flash Player appelle la méthode `runtimeAssetsLoadComplete()`.

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

Dans cette méthode, la variable rl représente le fichier SWF chargé. Le code appelle la méthode `getAssets()` du fichier SWF chargé, obtient la liste des éléments disponibles et remplit un composant ComboBox avec cette liste en appelant la méthode `populateDropDown()`. Cette méthode enregistre le chemin de classe complet de chaque élément. Si l'utilisateur clique sur le bouton Ajouter, l'interface déclenche la méthode `addAsset()` :

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

qui obtient le chemin de classe de l'élément actuellement sélectionné dans la ComboBox (`assetNameCbo.selectedItem.data`), et utilise la fonction `getDefinitionByName()` (du package flash.utils) pour obtenir une référence à la classe de l'élément afin de créer une nouvelle occurrence de celui-ci.

Chapitre 17 : Utilisation des interpolations de mouvement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

La section « Animation des objets » à la page 200 décrit la procédure d'implémentation d'une animation pilotée par un script ActionScript.

Ce chapitre est consacré à une autre technique de création d'une animation : l'interpolation de mouvement. Cette technique vous permet de créer un mouvement en le configurant en mode interactif dans un document par le biais d'Adobe® Flash® Professional. Vous pouvez ensuite utiliser ce mouvement dans votre animation dynamique à base de code ActionScript lors de l'exécution.

Flash Professional génère automatiquement le code ActionScript qui implémente l'interpolation de mouvement et vous permet de le copier et de le réutiliser.

Pour créer des interpolations de mouvement, vous devez disposer d'une licence pour Flash Professional.

Voir aussi

Package fl.motion

Principes de base des interpolations de mouvement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Les interpolations de mouvement permettent de créer aisément une animation.

Une interpolation de mouvement modifie les propriétés d'un objet d'affichage, telles que la position ou la rotation, d'une image à l'autre. Elle permet également de modifier l'apparence d'un objet d'affichage en cours de déplacement en lui appliquant divers filtres et propriétés. Vous créez l'interpolation de mouvement en mode interactif dans Flash Professional, ce qui génère le code ActionScript correspondant. Dans Flash, utilisez la commande Copie de mouvement en tant qu'ActionScript 3.0 pour copier le code ActionScript à l'origine de l'interpolation de mouvement. Vous pouvez alors réutiliser le code ActionScript pour créer un mouvement dans votre animation dynamique lors de l'exécution.

Pour plus d'informations sur la création d'une interpolation de mouvement, voir la section Interpolations de mouvement du manuel *Utilisation de Flash Professional*.

Concepts importants et terminologie

La liste de référence suivante contient un terme important relatif à la fonctionnalité étudiée :

Interpolation de mouvement Construction qui génère des images intermédiaires d'un objet d'affichage dans des états et à des moments différents, créant ainsi l'impression que le premier état évolue progressivement vers le second état. Une interpolation de mouvement permet de déplacer un objet d'affichage sur la scène, ainsi que de le faire progressivement grandir, rétrécir, pivoter ou changer de couleur.

Copie de scripts d'interpolation de mouvement dans Flash

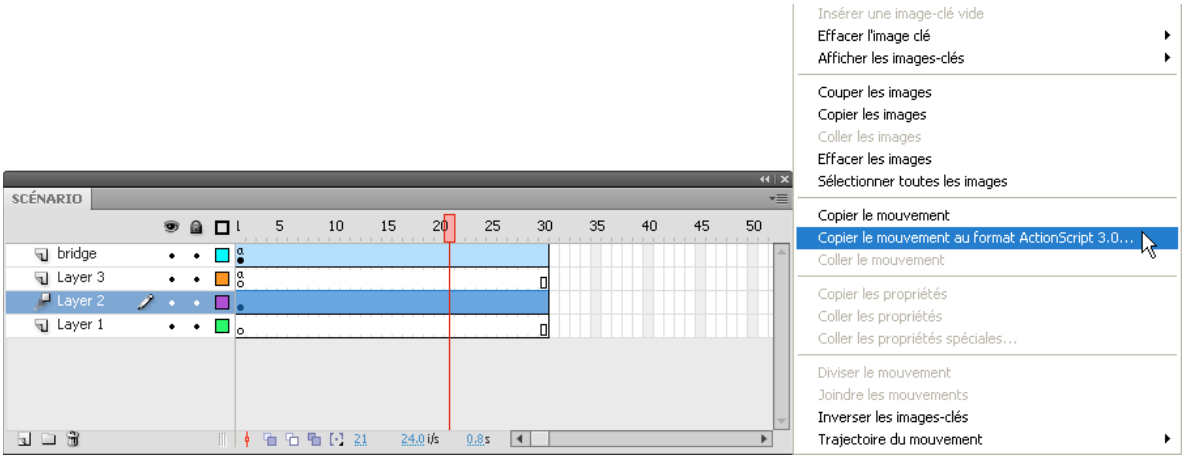
Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Une interpolation génère des images intermédiaires qui affichent un objet d'affichage dans des états différents dans deux images distinctes d'un scénario. Elle crée l'impression que le contenu de la première image se transforme progressivement en contenu de la seconde image. Dans une interpolation de mouvement, le changement d'apparence implique généralement la modification de la position de l'objet d'affichage, créant ainsi un mouvement. Outre le repositionnement de l'objet d'affichage, une interpolation de mouvement peut faire pivoter, incliner ou redimensionner ce dernier, voire lui appliquer des filtres.

Vous créez une interpolation de mouvement dans Flash en déplaçant un objet d'affichage entre des images-clé du scénario. Flash génère automatiquement le code ActionScript qui décrit l'interpolation, que vous pouvez copier et enregistrer dans un fichier. Pour plus d'informations sur la création d'une interpolation de mouvement, voir la section Interpolations de mouvement du manuel *Utilisation de Flash Professional*.

Vous pouvez accéder à la copie de mouvement en tant que commande ActionScript 3.0 dans Flash, de deux façons différentes. La première consiste à utiliser un menu contextuel d'interpolation sur la scène, comme suit :

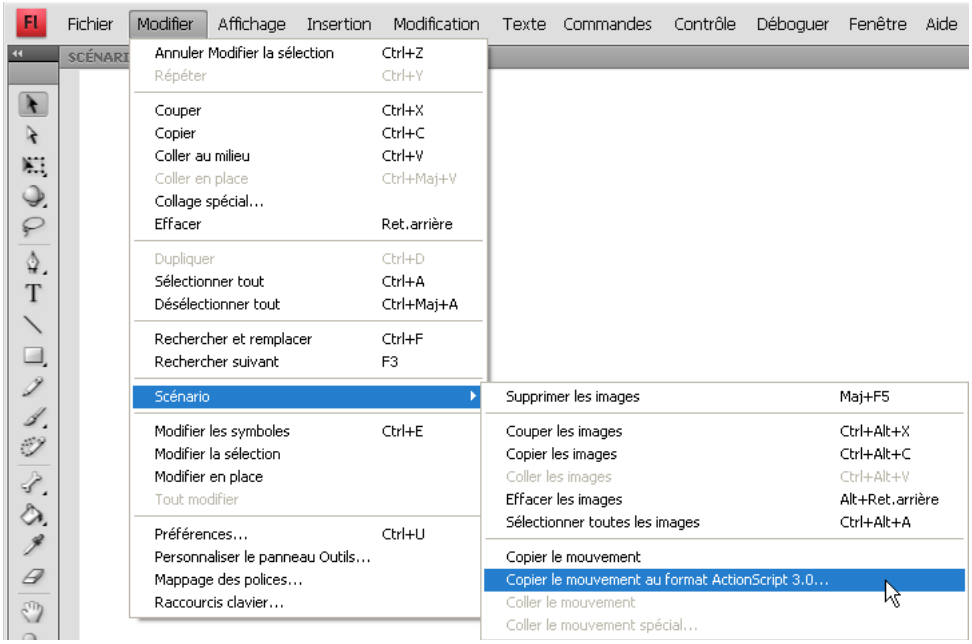
1. Sélectionnez l'interpolation de mouvement sur la scène.
2. Cliquez dessus avec le bouton droit de la souris (Windows) ou cliquez sur la touche Contrôle (Macintosh).
3. Choisissez Copie de mouvement en tant que ActionScript 3.0 . . .



La seconde technique consiste à choisir directement la commande dans le menu Edition de Flash, comme suit :

1. Sélectionnez l'interpolation de mouvement sur la scène.

2 Choisissez Edition > Scénario > Copier le mouvement en tant que ActionScript 3.0.



Une fois le script copié, collez-le dans un fichier et enregistrez-le.

Une fois l'interpolation de mouvement créée et après avoir copié et enregistré le script, vous pouvez la réutiliser sous sa forme actuelle ou la modifier dans votre propre animation dynamique pilotée par un script ActionScript.

Incorporation de scripts d'interpolation de mouvement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

L'en-tête du code ActionScript copié à partir de Flash recense tous les modules requis pour prendre en charge l'interpolation de mouvement.

Classes d'interpolation de mouvement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Les classes principales d'interpolation de mouvement, AnimatorFactory, MotionBase et Motion, appartiennent au package `fl.motion`. Vous pourriez avoir besoin de classes supplémentaires suivant les propriétés que l'interpolation de mouvement manipule. Par exemple, si l'interpolation de mouvement transforme ou fait pivoter l'objet d'affichage, vous devez importer les classes `flash.geom` appropriées. Si des filtres sont appliqués, importez les classes `flash.filter`. Dans ActionScript, une interpolation de mouvement est une occurrence de la classe Motion. La classe Motion stocke une séquence d'animation d'images-clés pouvant s'appliquer à un objet visuel. Les données de l'animation comprennent les éléments suivants : position, échelle, rotation, inclinaison, couleur, filtres et accélération.

Le code ActionScript suivant a été copié à partir d'une interpolation de mouvement créée dans Flash en vue d'animer un objet d'affichage portant le nom d'occurrence `Symbol1_2`. Il déclare une variable associée à un objet MotionBase nommé `__motion_Symbol1_2`. La classe MotionBase est le parent de la classe Motion.

```
var __motion_Symbol1_2:MotionBase;
```

Le script crée ensuite l'objet Motion :

```
__motion_Symbol1_2 = new Motion();
```

Noms d'objet Motion

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Dans le cas précédent, Flash a automatiquement généré le nom `__motion_Symbol1_2` de l'objet Motion. Il a associé le préfixe `__motion_` au nom de l'objet d'affichage. Ainsi, le nom généré automatiquement est basé sur le nom d'occurrence de l'objet cible de l'interpolation de mouvement dans Flash. La propriété `duration` de l'objet Motion indique le nombre total d'images dans l'interpolation de mouvement :

```
__motion_Symbol1_2.duration = 200;
```

Par défaut, Flash affecte automatiquement un nom à l'occurrence d'objet d'affichage dont l'interpolation de mouvement est copiée si elle n'en possède pas encore un.

Lorsque vous réutilisez un code ActionScript créé par Flash dans votre propre animation, vous pouvez conserver le nom d'interpolation automatiquement généré par Flash ou choisir un autre nom. Si vous modifiez le nom de l'interpolation, n'oubliez pas de répercuter la modification dans l'ensemble du script.

Dans Flash, vous pouvez également affecter le nom de votre choix à l'objet cible de l'interpolation de mouvement, puis créer l'interpolation de mouvement et copier le script. Quelle que soit l'approche adoptée, assurez-vous que chaque objet Motion de votre code ActionScript possède un nom unique.

Description de l'animation

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

La méthode `addPropertyArray()` de la classe MotionBase ajoute un tableau de valeurs pour décrire chaque propriété interpolée.

Le tableau contient potentiellement un élément par image-clé de l'interpolation de mouvement. Il arrive souvent que certains de ces tableaux contiennent moins d'éléments que le nombre total d'images-clés de l'interpolation de mouvement. Ce cas de figure se produit lorsque la dernière valeur du tableau n'est pas modifiée pour les images restantes.

Si la longueur de l'argument array est supérieure à celle de la propriété `duration` de l'objet Motion, la méthode `addPropertyArray()` ajuste en conséquence la valeur de la propriété `duration`. Elle n'ajoute pas d'image-clé pour les propriétés précédemment ajoutées. Les nouvelles images-clés subsistent pendant la durée des images supplémentaires de l'animation.

Les propriétés `x` et `y` de l'objet Motion décrivent la nouvelle position de l'objet interpolé au fur et à mesure de l'exécution de l'animation. Ces coordonnées correspondent aux valeurs les plus susceptibles de changer dans chaque image-clé si la position de l'objet d'affichage évolue. La méthode `addPropertyArray()` vous permet d'ajouter d'autres propriétés de mouvement. Par exemple, ajoutez les valeurs `scaleX` et `scaleY` si l'objet interpolé est redimensionné. Ajoutez les valeurs `skewX` et `skewY` s'il est incliné. Ajoutez la propriété `rotationConcat` s'il fait l'objet d'une rotation.

Utilisez la méthode `addPropertyArray()` pour définir les propriétés d'interpolation suivantes :

`x`	Position horizontale du point de transformation de l'objet dans l'espace de coordonnées de son objet parent
`y`	Position verticale du point de transformation de l'objet dans l'espace de coordonnées de son objet parent
`z`	Position de profondeur (axe z) du point de transformation de l'objet dans l'espace de coordonnées de son objet parent
`scaleX`	Redimensionnement horizontal, exprimé sous forme de pourcentage de l'objet tel qu'il est appliqué à partir du point de transformation
`scaleY`	Redimensionnement vertical, exprimé sous forme de pourcentage de l'objet tel qu'il est appliqué à partir du point de transformation
`skewX`	Angle d'inclinaison horizontale de l'objet, en degrés, tel qu'il est appliqué à partir du point de transformation
`skewY`	Angle d'inclinaison verticale de l'objet, en degrés, tel qu'il est appliqué à partir du point de transformation
`rotationX`	Rotation de l'objet autour de l'axe x à partir de son orientation d'origine
`rotationY`	Rotation de l'objet autour de l'axe y à partir de son orientation d'origine
`rotationConcat`	Valeurs de rotation (axe z) de l'objet dans le cadre du mouvement par rapport à l'orientation précédente appliquée à partir du point de transformation
`useRotationConcat`	Si cette propriété est définie, elle entraîne la rotation de l'objet cible lorsque la méthode `addPropertyArray()` fournit des données de mouvement.
`blendMode`	Valeur de la classe BlendMode qui définit le mélange de couleurs de l'objet sous lequel figurent les graphiques
`matrix3D`	Propriété matrix3D si elle a été définie pour l'image-clé. Réservée aux interpolations 3D. Si elle est utilisée, aucune des propriétés de transformation précédentes n'est prise en considération.
`rotationZ`	Rotation (en degrés) autour de l'axe z de l'objet, à partir de son orientation d'origine par rapport au conteneur parent 3D. Utilisé pour les interpolations 3D au lieu de rotationConcat.

Les propriétés ajoutées au script automatiquement généré varient selon les propriétés affectées à l'interpolation de mouvement dans Flash. Vous pouvez ajouter, supprimer ou modifier certaines de ces propriétés lorsque vous personnalisez votre version du script.

Le code suivant affecte des valeurs aux propriétés de l'interpolation de mouvement `__motion_Wheel`. Dans ce cas de figure, l'objet d'affichage interpolé ne change pas d'emplacement, mais pivote sur place dans les 29 images de l'interpolation de mouvement. Les diverses valeurs affectées au tableau `rotationConcat` définissent la rotation. Les autres valeurs de propriété de cette interpolation de mouvement ne sont pas modifiées.

```
__motion_Wheel = new Motion();
__motion_Wheel.duration = 29;
__motion_Wheel.addPropertyArray("x", [0]);
__motion_Wheel.addPropertyArray("y", [0]);
__motion_Wheel.addPropertyArray("scaleX", [1.00]);
__motion_Wheel.addPropertyArray("scaleY", [1.00]);
__motion_Wheel.addPropertyArray("skewX", [0]);
__motion_Wheel.addPropertyArray("skewY", [0]);
__motion_Wheel.addPropertyArray("rotationConcat",
    [
        0,-13.2143,-26.4285,-39.6428,-52.8571,-66.0714,-79.2857,-92.4999,-105.714,
        -118.929,-132.143,-145.357,-158.571,-171.786,-185,-198.214,-211.429,-224.643,
        -237.857,-251.071,-264.286,-277.5,-290.714,-303.929,-317.143,-330.357,
        -343.571,-356.786,-370
    ]
);
__motion_Wheel.addPropertyArray("blendMode", ["normal"]);
```

Dans l'exemple suivant, l'objet d'affichage `Leaf_1` traverse la scène. Les tableaux de propriétés `x` et `y` correspondants contiennent des valeurs différentes pour chacune des 100 images de l'animation. Par ailleurs, l'objet pivote sur son axe `z` au fur et à mesure qu'il traverse la scène. Les divers éléments du tableau de la propriété `rotationZ` déterminent la rotation.

```
__motion_Leaf_1 = new MotionBase();
__motion_Leaf_1.duration = 100;
__motion_Symbol1_4.addPropertyArray("y",
    [
        0,5.91999,11.84,17.76,23.68,29.6,35.52,41.44,47.36,53.28,59.2,65.12,71.04,
        76.96,82.88,88.8,94.72,100.64,106.56,112.48,118.4,124.32,130.24,136.16,142.08,
        148,150.455,152.909,155.364,157.818,160.273,162.727,165.182,167.636,170.091,
        172.545,175,177.455,179.909,182.364,184.818,187.273,189.727,192.182,194.636,
        197.091,199.545,202,207.433,212.865,218.298,223.73,229.163,234.596,240.028,
        245.461,250.893,256.326,261.759,267.191,272.624,278.057,283.489,
        288.922,294.354,299.787,305.22,310.652,316.085,321.517,326.95,330.475,334,
        337.525,341.05,344.575,348.1,351.625,355.15,358.675,362.2,365.725,369.25,
        372.775,376.3,379.825,383.35,386.875,390.4,393.925,397.45,400.975,404.5,
        407.5,410.5,413.5,416.5,419.5,422.5,425.5
    ]
);
__motion_Symbol1_4.addPropertyArray("scaleX", [1.00]);
__motion_Symbol1_4.addPropertyArray("scaleY", [1.00]);
__motion_Symbol1_4.addPropertyArray("skewX", [0]);
__motion_Symbol1_4.addPropertyArray("skewY", [0]);
__motion_Symbol1_4.addPropertyArray("z",
    [
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ]
);
__motion_Symbol1_4.addPropertyArray("rotationX", [64.0361]);
__motion_Symbol1_4.addPropertyArray("rotationY", [41.9578]);
__motion_Symbol1_4.addPropertyArray("rotationZ",
    [
        -18.0336,-17.5536,-17.0736,-16.5936,-16.1136,-15.6336,-15.1536,-14.6736,
        -14.1936,-13.7136,-13.2336,-12.7536,-12.2736,-11.7936,-11.3136,-10.8336,
        -10.3536,-9.8736,-9.3936,-8.9136,-8.4336,-7.9536,-7.4736,-6.9936,-6.5136,
        -6.0336,-7.21542,-8.39723,-9.57905,-10.7609,-11.9427,-13.1245,-14.3063,
        -15.4881,-16.67,-17.8518,-19.0336,-20.2154,-21.3972,-22.5791,-23.7609,
        -24.9427,-26.1245,-27.3063,-28.4881,-29.67,-30.8518,-32.0336,-31.0771,
        -30.1206,-29.164,-28.2075,-27.251,-26.2945,-25.338,-24.3814,-23.4249,
        -22.4684,-21.5119,-20.5553,-19.5988,-18.6423,-17.6858,-16.7293,-15.7727
        -14.8162,-13.8597,-12.9032,-11.9466,-10.9901,-10.0336,-10.9427,-11.8518,
        -12.7609,-13.67,-14.5791,-15.4881,-16.3972,-17.3063,-18.2154,-19.1245,
        -20.0336,-20.9427,-21.8518,-22.7609,-23.67,-24.5791,-25.4881,-26.3972,
        -27.3063,-28.2154,-29.1245,-30.0336,-28.3193,-26.605,-24.8907,-23.1765,
        -21.4622,-19.7479,-18.0336
    ]
);
__motion_Symbol1_4.addPropertyArray("blendMode", ["normal"]);
```

Ajout de filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Si l'objet cible d'une interpolation de mouvement contient des filtres, ces derniers sont ajoutés par le biais des méthodes `initFilters()` et `addFilterPropertyArray()` de la classe Motion.

Initialisation du tableau de filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

La méthode `initFilters()` initialise les filtres. Son premier argument est un tableau qui recense les noms de classe entièrement qualifiés de tous les filtres appliqués à l'objet d'affichage. Ce tableau de noms de filtre est généré à partir de la liste de filtres associée à l'interpolation de mouvement dans Flash. Dans votre copie du script, vous pouvez supprimer ou ajouter dans ce tableau n'importe quel filtre du package `flash.filters`. L'appel suivant initialise la liste de filtres associée à l'objet d'affichage cible. Il applique les filtres `DropShadowFilter`, `GlowFilter` et `BevelFilter` et copie la liste dans chaque image-clé de l'objet Motion.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

Ajout de filtres

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

La méthode `addFilterPropertyArray()` décrit les propriétés d'un filtre initialisé doté des arguments suivants :

1. Son premier argument identifie un filtre en fonction de son index. L'index se réfère à la position du nom de filtre dans le tableau des noms de classe de filtre transmis lors d'un appel précédent d'`initFilters()`.
2. Le deuxième argument est la propriété à stocker pour le filtre dans chaque image-clé.
3. Le troisième argument est la valeur de la propriété de filtre indiquée.

Etant donné l'appel précédent d'`initFilters()`, les appels suivants de `addFilterPropertyArray()` affectent la valeur 5 aux propriétés `blurX` et `blurY` de `DropShadowFilter`. `DropShadowFilter` est le premier élément (index 0)du tableau de filtres initialisés :

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

Les trois appels suivants affectent des valeurs aux propriétés qualité, alpha et couleur de `GlowFilter` (le deuxième élément (index 1) du tableau de filtres initialisés) :

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

Les quatre appels suivants affectent des valeurs aux propriétés `shadowAlpha`, `shadowColor`, `highlightAlpha` et `highlightColor` de `BevelFilter`, le troisième élément (index 2) du tableau de filtres initialisés :

```
__motion_Box.addFilterPropertyArray(2, "shadowAlpha", [1.00]);
__motion_Box.addFilterPropertyArray(2, "shadowColor", [0x000000]);
__motion_Box.addFilterPropertyArray(2, "highlightAlpha", [1.00]);
__motion_Box.addFilterPropertyArray(2, "highlightColor", [0xffffff]);
```

Réglage de la couleur à l'aide de ColorMatrixFilter

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

Une fois `ColorMatrixFilter` initialisé, vous pouvez définir les propriétés `AdjustColor` appropriées pour régler la luminosité, le contraste, la saturation et la teinte de l'objet d'affichage interpolé. En règle générale, le filtre `AdjustColor` est appliqué lors de la génération de l'interpolation de mouvement dans Flash. Vous pouvez l'ajuster dans votre copie du code ActionScript. L'exemple suivant transforme la teinte et la saturation de l'objet d'affichage au fur et à mesure qu'il se déplace.

```
__motion_Leaf_1.initFilters(["flash.filters.ColorMatrix"], [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorBrightness", [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorContrast", [0], -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorSaturation",
    [
        0,-0.589039,1.17808,-1.76712,-2.35616,-2.9452,-3.53424,-4.12328,
        -4.71232,-5.30136,-5.89041, 6.47945,-7.06849,-7.65753,-8.24657,
        -8.83561,-9.42465,-10.0137,-10.6027,-11.1918,11.7808,-12.3699,
        -12.9589,-13.5479,-14.137,-14.726,-15.3151,-15.9041,-16.4931,
        17.0822,-17.6712,-18.2603,-18.8493,-19.4383,-20.0274,-20.6164,
        -21.2055,-21.7945,22.3836,-22.9726,-23.5616,-24.1507,-24.7397,
        -25.3288,-25.9178,-26.5068,-27.0959,27.6849,-28.274,-28.863,-29.452,
        -30.0411,-30.6301,-31.2192,-31.8082,-32.3973,32.9863,-33.5753,
        -34.1644,-34.7534,-35.3425,-35.9315,-36.5205,-37.1096,-37.6986,
        38.2877,-38.8767,-39.4657,-40.0548,-40.6438,-41.2329,-41.8219,
        -42.411,-43
    ],
    -1, -1);
__motion_Leaf_1.addFilterPropertyArray(0, "adjustColorHue",
    [
        0,0.677418,1.35484,2.03226,2.70967,3.38709,4.06451,4.74193,5.41935,
        6.09677,6.77419,7.45161,8.12903,8.80645,9.48387,10.1613,10.8387,11.5161,
        12.1935,12.871,13.5484,14.2258,14.9032,15.5806,16.2581,16.9355,17.6129,
        18.2903,18.9677,19.6452,20.3226,21,22.4286,23.8571,25.2857,26.7143,28.1429,
        29.5714,31,32.4286,33.8571,35.2857,36.7143,38.1429,39.5714,41,42.4286,43.8571,
        45.2857,46.7143,48.1429,49.5714,51,54,57,60,63,66,69,72,75,78,81,84,87,
        90,93,96,99,102,105,108,111,114
    ],
    -1, -1);
```

Association d'une interpolation de mouvement à ses objets d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures, Flash CS3 ou ultérieur requis

La dernière tâche consiste à associer l'interpolation de mouvement aux objets d'affichage qu'elle manipule.

La classe AnimatorFactory gère l'association entre une interpolation de mouvement et ses objets d'affichage cible. L'argument du constructeur d'AnimatorFactory correspond à l'objet Motion :

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

La méthode `addTarget()` de la classe AnimatorFactory permet d'associer l'objet d'affichage cible à l'interpolation de mouvement correspondante. Le code ActionScript copié à partir de Flash met la ligne `addTarget()` en commentaire et n'indique pas de nom d'occurrence :

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

Dans votre copie, stipulez l'objet d'affichage à associer à l'interpolation de mouvement. Dans l'exemple suivant, les cibles spécifiées sont `greenWheel` et `redWheel` :

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

Vous pouvez associer plusieurs objets d'affichage à la même interpolation de mouvement en utilisant plusieurs appels de `addTarget()`.

Chapitre 18 : Utilisation de la cinématique inverse

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

La cinématique inverse (IK, Inverse kinematics) est une technique de création fantastique de mouvement réaliste.

IK vous permet de créer des mouvements coordonnés au sein d'une chaîne de sections connectées appelée squelette IK, de sorte que les sections se déplacent avec réalisme. Les sections du squelette représentent ses os et articulations. A partir de l'extrémité finale du squelette, IK calcule les angles des articulations requises pour atteindre cette dernière.

Calculer manuellement ces angles s'avérerait particulièrement complexe. Cette fonctionnalité présente l'avantage de permettre de créer des squelettes en mode interactif dans Adobe® Flash® Professional. Il vous suffit ensuite de les animer par le biais d'ActionScript. Le moteur IK intégré à Flash Professional exécute les calculs requis pour décrire le mouvement du squelette. Vous pouvez restreindre le mouvement à certains paramètres dans votre code ActionScript.

La version Flash Professional CS5 de la cinématique inverse (IK) intègre à présent le concept de ressort de segment, généralement réservé aux applications d'animation haut de gamme. Associée au nouveau moteur physique dynamique, cette fonctionnalité permet de configurer des mouvements réalistes. Cet effet est par ailleurs visible lors des phases d'exécution et de création.

Pour créer des squelettes de cinématique inverse, vous devez disposer d'une licence pour Flash Professional.

Voir aussi

Package fl.ik

Principes de base de la cinématique inverse

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

La cinématique inverse (IK) vous permet de créer une animation réaliste en liant des sections de sorte qu'elles se déplacent les unes par rapport aux autres avec naturel.

L'utilisation d'IK vous permet par exemple de déplacer une jambe pour qu'elle occupe une position déterminée en articulant les mouvements des articulations de la jambe nécessaires à l'obtention de la pose appropriée. IK utilise une structure osseuse chaînée portant le nom de squelette IK. Le package `fl.ik` vous aide à créer des animations qui ressemblent à un mouvement naturel. Il vous permet d'animer plusieurs squelettes IK en toute transparence sans avoir à maîtriser les concepts physiques sur lesquels s'appuient les algorithmes IK.

Vous créez le squelette IK et les os et articulations qui le composent dans Flash Professional. Vous pouvez ensuite utiliser les classes IK pour les animer lors de l'exécution.

Pour obtenir des instructions détaillées sur la création d'un squelette IK, voir Utilisation de la cinématique inverse dans *Utilisation de Flash Professional*.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la fonctionnalité étudiée.

Squelette Chaîne cinématique composée d'os et d'articulations, utilisée en animation informatique pour simuler un mouvement réaliste.

Os Segment rigide d'un squelette, équivalent à un os chez un animal.

Cinématique inverse (IK) Processus d'identification des paramètres d'un objet souple articulé appelé squelette ou chaîne cinématique.

Articulation Emplacement où deux os s'unissent, conçu pour permettre le mouvement des os ; analogue à une articulation chez un animal.

Moteur physique Package d'algorithmes physiques qui permet d'intégrer des actions réalistes à l'animation.

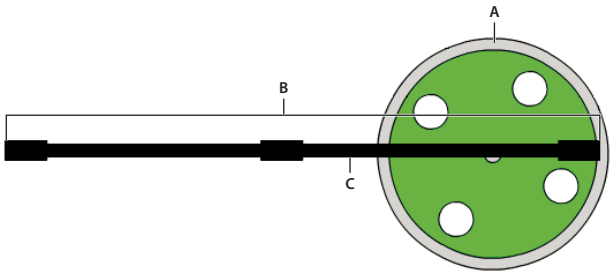
Ressort Qualité d'un segment qui se déplace et réagit lorsque le segment parent est déplacé, puis diminue progressivement par incréments.

Aperçu de l'animation de squelettes IK

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

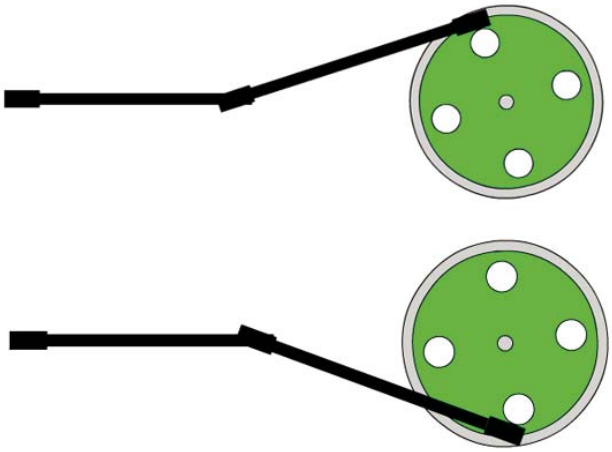
Une fois le squelette IK créé dans Flash Professional, utilisez les classes `fl.ik` pour restreindre ses mouvements, suivre les événements correspondants et l'animer lors de l'exécution.

La figure ci-dessous illustre le clip `Wheel`. L'essieu est une occurrence d'un squelette IK appelée `Axle`. La classe IKMover déplace le squelette en le synchronisant avec la rotation d'une roue. Dans le squelette, IKBone, nommé `ikBone2`, est rattaché à la roue au niveau de l'articulation arrière.



***A.** Roue **B.** Essieu **C.** ikBone2*

Lors de l'exécution, la roue tourne en association avec l'interpolation de mouvement `__motion_Wheel` étudiée dans « Description de l'animation » à la page 349. Un objet IKMover lance et contrôle le mouvement de l'essieu. La figure suivante propose deux instantanés du squelette de l'essieu connecté à la roue qui tourne sur différentes images de la rotation.



Lors de l'exécution, le code ActionScript suivant :

- Extrait des informations relatives au squelette et à ses composants.
- Instancie un objet IKMover.
- Déplace l'essieu en conjonction avec la rotation de la roue.

```
import fl.ik.*

var tree:IKArmature = IKManager.getArmatureByName("Axle");
var bone:IKBone = tree.getBoneByName("ikBone2");
var endEffector:IKJoint = bone.tailJoint;
var pos:Point = endEffector.position;

var ik:IKMover = new IKMover(endEffector, pos);
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;

Wheel.addEventListener(Event.ENTER_FRAME, frameFunc);

function frameFunc(event:Event)
{
    if (Wheel != null)
    {
        var mat:Matrix = Wheel.transform.matrix;
        var pt = new Point(90, 0);
        pt = mat.transformPoint(pt);

        ik.moveTo(pt);
    }
}
```

Les classes IK utilisées pour déplacer l'essieu sont les suivantes :

- IKArmature : décrit le squelette (structure arborescente composée d'os et d'articulations). A créer dans Flash Professional.
- IKManager : classe qui contient tous les squelettes IK du document, à créer dans Flash Professional.
- IKBone : segment d'un squelette IK.
- IKJoint : connexion entre deux os IK.
- IKMover : lance et contrôle le mouvement IK des squelettes.

Pour obtenir une description détaillée de ces classes, voir ik package (Package ik).

Obtention d'informations sur un squelette IK

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

Commencez par déclarer les variables associées au squelette, à l'os et à l'articulation qui composent les sections à déplacer.

Le code suivant utilise la méthode `getsqueletteByName()` de la classe IKManager pour affecter la valeur du squelette Axle à la variable IKArmature `tree`. Le squelette Axle a été précédemment généré dans Flash Professional.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

De même, le code suivant utilise la méthode `getBoneByName()` de la classe IKArmature pour affecter à la variable IKBone la valeur de l'os `ikBone2`.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

L'articulation arrière de l'os `ikBone2` correspond à la section du squelette connectée à la roue qui tourne.

La ligne suivante déclare la variable `endEffector` et l'affecte à la propriété `tailjoint` de l'os `ikBone2` :

```
var endEffector:IKJoint = home.tailjoint;
```

La variable `pos` est un point qui stocke la position actuelle de l'articulation `endEffector`.

```
var pos:Point = endEffector.position;
```

Dans cet exemple, `pos` correspond à la position de l'articulation raccordée à la roue à l'extrémité de l'essieu. La valeur d'origine de cette variable est extraite de la propriété `position` de IKJoint.

Instanciation de l'objet IKMover et restriction du mouvement

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

Une occurrence de la classe IKMover déplace l'essieu.

La ligne suivante instancie l'objet IKMover `ik` et transmet à son constructeur l'élément à déplacer et le point de départ du mouvement :

```
var ik:IKMover = new IKMover(endEffector, pos);
```

Les propriétés de la classe IKMover vous permettent de restreindre le mouvement d'un squelette. Vous pouvez restreindre le mouvement en fonction de la distance, des itérations et de la durée du mouvement.

Les paires de propriétés suivantes imposent ces restrictions. Les paires se composent d'une valeur booléenne qui indique si le mouvement est restreint et d'un entier stipulant la restriction :

Propriété Boolean	Propriété Integer	Restriction définie
`limitByDistance:Boolean`	`distanceLimit:int`	Définit la distance maximale en pixels parcourue par le moteur IK par itération.
`limitByIteration:Boolean`	`iterationLimit:int`	Définit le nombre maximal d'itérations effectuées par le moteur IK par mouvement.
`limitByTime:Boolean`	`timeLimit:int`	Définit la durée maximale, exprimée en millisecondes, affectée au moteur IK pour effectuer le mouvement.

Toutes les valeurs booléennes étant définies sur `false` par défaut, le mouvement n'est pas restreint, sauf si vous avez explicitement défini une valeur booléenne sur `true`. Pour imposer une restriction, définissez la propriété appropriée sur `true`, puis indiquez la valeur de la propriété Integer correspondante. Si vous définissez la restriction sur une valeur sans définir la propriété Boolean correspondante, elle n'est pas prise en considération. Dans ce cas, le moteur IK continue à déplacer l'objet jusqu'à ce qu'une autre restriction ou la position cible de l'objet IKMover soit atteinte.

Dans l'exemple suivant, la distance maximale parcourue par le mouvement du squelette est définie sur 0,1 pixel par itération. Le nombre maximal d'itérations par mouvement est défini sur dix.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

Mouvement d'un squelette IK

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

L'objet IKMover déplace l'essieu au sein de l'écouteur d'événement associé à la roue. A chaque événement enterFrame de la roue, une nouvelle position cible du squelette est calculée. Par le biais de sa méthode `moveTo()`, l'objet IKMover place l'articulation arrière sur sa position cible ou parcourt une distance aussi longue que possible sans enfreindre les contraintes définies par ses propriétés `limitByDistance`, `limitByIteration` et `limitByTime`.

```
Wheel.addEventListener(Event.ENTER_FRAME, frameFunc);

function frameFunc(event:Event)
{
    if (Wheel != null)
    {
        var mat:Matrix = Wheel.transform.matrix;
        var pt = new Point(90,0);
        pt = mat.transformPoint(pt);

        ik.moveTo(pt);
    }
}
```

Utilisation de ressorts

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS5 ou ultérieur requis

La cinématique inverse prend en charge le ressort de segment dans Flash Professional CS5. Vous pouvez définir les ressorts de segment lors de la phase de création et ajouter ou modifier les attributs correspondants lors de l'exécution. La propriété Spring se rapporte à un segment et aux liaisons correspondantes. Elle possède deux attributs, `IKJoint.springStrength`, qui définit l'intensité du ressort, et `IKJoint.springDamping`, qui ajoute une résistance à la valeur d'intensité et modifie la valeur de décroissance du ressort.

L'intensité du ressort est exprimée sous forme de pourcentage compris entre la valeur par défaut, 0 (rigidité absolue) et 100 (élasticité importante contrôlée par les lois de la physique). Les segments à ressort réagissent au mouvement de la liaison correspondante. Si aucune autre translation (rotation, x ou y) n'est activée, les paramètres du ressort n'ont aucun impact.

L'amortissement du ressort est exprimé sous forme de pourcentage, compris entre la valeur par défaut, 0 (aucune résistance) et 100 (amortissement important). L'amortissement modifie la durée de l'intervalle qui sépare le mouvement initial d'un segment et son retour à une position de repos.

Vérifiez si des ressorts sont associés à un objet IKArmature par le biais de la propriété `IKArmature.springsEnabled`. Les autres propriétés et méthodes relatives aux ressorts relèvent d'objets IKJoint individuels. Une liaison peut être soumise à une rotation angulaire et à une translation le long des axes x et y. Faites appel à `IKJoint.setSpringAngle` pour positionner l'angle de rotation d'un ressort de liaison et à `IKJoint.setSpringPt` pour définir la position par translation d'un ressort de liaison.

Cet exemple sélectionne un segment par nom et identifie la propriété tailJoint correspondante. Le code teste le squelette parent pour vérifier si des ressorts sont activés, puis définit les propriétés du ressort de la liaison.

```
var arm:IKArmature = IKManager.getArmatureAt(0);
var bone:IKBone = arm.getBoneByName("c");
var joint:IKJoint = bone.tailJoint;
if (arm.springsEnabled) {
    joint.springStrength = 50; //medium spring strength
    joint.springDamping = 10; //light damping resistance
    if (joint.hasSpringAngle) {
        joint.setSpringAngle(30); //set angle for rotational spring
    }
}
```

Utilisation d'événements IK

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures, Flash CS4 ou ultérieur requis

La classe IKEvent vous permet de créer un objet événement qui contient des informations sur les événements IK. Une information IKEvent décrit le mouvement qui s'est arrêté car la durée, la distance ou le nombre maximal d'itérations stipulés ont été dépassés.

Le code suivant indique un écouteur et un gestionnaire d'événement destinés au suivi des événements de limite de temps. Ce gestionnaire d'événement signale les propriétés de durée, distance, nombre d'itérations et articulations d'un événement déclenché lorsque la durée maximale de l'objet IKMover est dépassée.

```
var ikmover:IKMover = new IKMover(endjoint, pos);
ikMover.limitByTime = true;
ikMover.timeLimit = 1000;

ikmover.addEventListener(IKEvent.TIME_LIMIT, timeLimitFunction);

function timeLimitFunction(evt:IKEvent):void
{
    trace("timeLimit hit");
    trace("time is " + evt.time);
    trace("distance is " + evt.distance);
    trace("iterationCount is " + evt.iterationCount);
    trace("IKJoint is " + evt.joint.name);
}
```

Chapitre 19 : Travail en trois dimensions (3D)

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les moteurs d'exécution de Flash Player et d'AIR prennent en charge les graphiques 3D de deux manières. Vous pouvez utiliser les objets d'affichage tridimensionnels sur la liste d'affichage de Flash. Cette méthode permet d'ajouter des effets tridimensionnels à du contenu Flash et convient aux objets comportant peu de polygones. Dans Flash Player 11 et AIR 3 ou les versions ultérieures, vous pouvez effectuer le rendu de séquences 3D complexes à l'aide de l'API Stage3D.

Une fenêtre d'affichage Stage3D n'est pas un objet d'affichage. Les graphiques 3D sont rendus dans une fenêtre d'affichage qui apparaît sous la liste d'affichage de Flash (et au-dessus de tous les plans de fenêtre d'affichage StageVideo). Plutôt que d'utiliser les classes DisplayObject de Flash pour créer une séquence, utilisez un processus 3D programmable (similaire à OpenGL et Direct3D). Ce processus prend les données et les textures comme entrées et effectue le rendu de la séquence à l'aide des programmes de shaders que vous fournissez. L'accélération matérielle est utilisée lorsqu'un processeur graphique (GPU) compatible disposant des pilotes pris en charge est disponible sur l'ordinateur client.

Stage3D fournit une API de très bas niveau. Dans une application, vous êtes encouragé à utiliser une structure d'application 3D prenant en charge Stage3D. Vous pouvez créer votre propre structure d'application, ou utiliser l'une des nombreuses structures commerciales et Open Source déjà disponibles.

Pour plus d'informations sur le développement d'applications 3D à l'aide de Stage3D et sur les structures d'application 3D disponibles, voir Flash Player Developer Center: Stage 3D.

Principes de base des objets d'affichage 3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La principale différence entre un objet en deux dimensions (2D) et un objet en trois dimensions (3D) projeté sur un écran en deux dimensions consiste en l'ajout d'une troisième dimension à l'objet. La troisième dimension permet à l'objet de se rapprocher et de s'éloigner du point de vue de l'utilisateur.

Lorsque vous définissez explicitement la propriété `z` d'un objet d'affichage sur une valeur numérique, l'objet crée automatiquement une matrice de transformation 3D. Vous pouvez intervenir sur cette matrice pour modifier les paramètres de transformation 3D de l'objet.

En outre, la rotation 3D diffère de la rotation 2D. En 2D, l'axe de la rotation est systématiquement perpendiculaire au plan x/y, autrement dit, elle se trouve sur l'axe z. En 3D, l'axe de rotation peut se trouver autour de n'importe lequel des axes, x, y ou z. La définition des propriétés de rotation et de mise à l'échelle d'un objet d'affichage lui permet de se déplacer dans l'espace 3D.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants utilisés dans le cadre de la programmation de graphiques en 3 dimensions :

Perspective Dans un plan 2D, représentation de lignes parallèles convergeant vers un point de fuite pour donner une illusion de profondeur et de distance.

Projection Génération d'une image 2D d'un objet 3D. La projection 3D mappe des points 3D sur un plan 2D.

Rotation Modification de l'orientation (et souvent de la position) d'un objet en faisant décrire un mouvement circulaire à chacun de ses points.

Transformation Modification de points 3D ou d'ensemble de points par translation, rotation, mise à l'échelle, inclinaison ou une combinaison de ces actions.

Translation Modification de la position d'un objet en déplaçant chacun de ses points sur une distance et dans une direction identiques.

Point de fuite Point auquel des lignes parallèles qui s'éloignent semblent se rencontrer lorsqu'elles sont représentées dans une perspective linéaire.

Vecteur Un vecteur 3D représente un point ou un emplacement dans l'espace en trois dimensions à l'aide de coordonnées cartésiennes (x,y,z).

Sommet Point.

Maillage texturé Tout point définissant un objet dans l'espace 3D.

Mappage des coordonnées UV Mode d'application d'une texture ou d'une image bitmap à une surface 3D. Le mappage des coordonnées UV affecte des valeurs à des coordonnées sur une image en tant que pourcentages de l'axe horizontal (U) et de l'axe vertical (V).

valeur T Facteur de mise à l'échelle permettant de déterminer la taille d'un objet 3D lorsque celui-ci se rapproche ou s'éloigne du point de vue actuel.

Culling Rendu, ou non, des surfaces avec un enroulement spécifique. L'utilisation du culling (élimination) permet de masquer des surfaces qui ne sont pas visibles à partir du point de vue actuel.

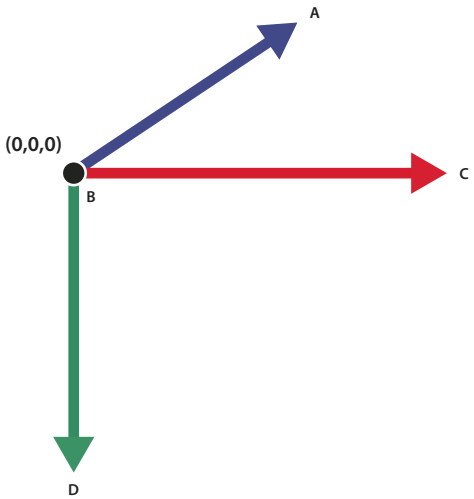
Présentation des objets d'affichage 3D dans les moteurs d'exécution de Flash Player et d'AIR

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Dans les versions de Flash Player antérieures à Flash Player 10 et dans les versions d'Adobe AIR antérieures à Adobe AIR 1.5, les objets d'affichage possèdent deux propriétés, `x` et `y`, permettant de positionner ces derniers sur un plan 2D. A partir de Flash Player 10 et Adobe AIR 1.5, tout objet d'affichage ActionScript est doté d'une propriété `z` permettant de le positionner le long de l'axe z, qui est généralement utilisé pour indiquer la profondeur ou la distance.

Flash Player 10 et Adobe AIR 1.5 prennent désormais en charge les effets 3D. Les objets 3D restent cependant essentiellement plats. Tout objet d'affichage, tel qu'un objet MovieClip ou Sprite, effectue en fait son propre rendu en deux dimensions, sur un plan unique. Les fonctions 3D vous permettent de placer, déplacer, faire pivoter et transformer de diverses façons ces objets planaires dans la totalité des trois dimensions. Elles vous permettent également de gérer les points 3D et de les convertir en coordonnées 2D (x et y), afin que vous puissiez projeter les objets 3D sur un affichage 2D. A l'aide de ces fonctions, vous pouvez simuler de nombreux types d'effets 3D.

Le système de coordonnées 3D utilisé par ActionScript est différent. Lorsque vous utilisez des coordonnées 2D dans ActionScript, la valeur de x augmente au fur et à mesure du déplacement vers la droite le long de l'axe x, et la valeur de y augmente au fur et à mesure du déplacement vers le bas le long de l'axe y. Le système de coordonnées 3D conserve ces conventions et ajoute un axe z dont la valeur augmente au fur et à mesure que vous vous éloignez du point de vue.



Directions positives des axes x, y et z dans le système de coordonnées 3D
***A.** + axe Z **B.** Origine **C.** + axe X **D.** + axe Y*

***Remarque :** n'oubliez pas que Flash Player et AIR représentent toujours la 3D en calques. Par conséquent, si l'objet A se trouve devant l'objet B dans la liste d'affichage, Flash Player ou AIR rend toujours A devant B, quelles que soient les valeurs sur l'axe z des deux objets. Pour résoudre le conflit entre l'ordre de la liste d'affichage et celui de l'axe z, utilisez la méthode* `transform.getRelativeMatrix3D()` *afin d'enregistrer, puis de réorganiser les calques des objets d'affichage 3D. pour plus d'informations, voir «* *Réorganisation de l'affichage à l'aide d'objets Matrix3D* » *à la page 374.*

Les classes ActionScript suivantes prennent en charge les nouvelles fonctions 3D :

1. La classe flash.display.DisplayObject contient la propriété `z` et de nouvelles propriétés de rotation et de mise à l'échelle permettant de manipuler les objets d'affichage dans l'espace 3D. La méthode `DisplayObject.local3DToGlobal()` simplifie la projection de géométrie 3D sur un plan 2D.
2. Vous pouvez utiliser la classe flash.geom.Vector3D en tant que structure de données pour la gestion des points 3D. Elle prend aussi en charge la mathématique vectorielle.
3. La classe flash.geom.Matrix3D prend en charge les transformations complexes de géométrie 3D, telles que la rotation, la mise à l'échelle et la translation.
4. La classe flash.geom.PerspectiveProjection contrôle les paramètres de mappage de géométrie 3D sur un affichage 2D.

ActionScript propose deux approches différentes pour simuler des images 3D :

1. Agencement et animation d'objets planaires dans l'espace 3D. Cette approche implique l'animation d'objets d'affichage à l'aide de leurs propriétés x, y et z, ou la définition des propriétés de rotation et de mise à l'échelle par le biais de la classe DisplayObject. Il est possible de générer des mouvements plus complexes à l'aide de l'objet DisplayObject.transform.matrix3D. L'objet DisplayObject.transform.perspectiveProjection personnalise le tracé des objets d'affichage dans la perspective 3D. Adoptez cette approche pour animer des objets 3D principalement composés de plans. Elle convient aux galeries d'image 3D ou aux objets d'animation 2D agencés dans l'espace 3D, par exemple.

2 Génération de triangles 2D à partir d'une géométrie 3D et rendu de ces triangles avec des textures. Pour ce faire, vous devez d'abord définir et gérer des données relatives aux objets 3D, puis les convertir en triangles 2D à des fins de rendu. Il est possible de mapper des textures bitmap sur ces triangles, qui sont ensuite tracés sur un objet graphique à l'aide de la méthode `Graphics.drawTriangles()`. Cette approche est appropriée pour le chargement des données d'un modèle 3D à partir d'un fichier et le rendu du modèle à l'écran, ou pour la génération et le tracé d'un terrain 3D en tant que maillages triangulaires, par exemple.

Création et déplacement d'objets d'affichage 3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Pour convertir un objet d'affichage 2D en objet d'affichage 3D, vous devez explicitement définir sa propriété `z` sur une valeur numérique. Lorsque vous affectez une valeur à la propriété `z`, un objet Transform est créé pour l'objet d'affichage. La définition des propriétés `DisplayObject.rotationX` ou `DisplayObject.rotationY` crée également un objet Transform. Celui-ci contient une propriété `Matrix3D` qui régit la représentation de l'objet d'affichage dans l'espace 3D.

Le code suivant définit les coordonnées d'un objet d'affichage appelé « leaf » (feuille) :

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

Vous pouvez visualiser ces valeurs, ainsi que les propriétés qui en dérivent, dans la propriété `matrix3D` de l'objet Transform de la feuille :

```
var leafMatrix:Matrix3D  = leaf.transform.matrix3D;

trace(leafMatrix.position.x);
trace(leafMatrix.position.y);
trace(leafMatrix.position.z);
trace(leafMatrix.position.length);
trace(leafMatrix.position.lengthSquared);
```

Pour plus d'informations sur les propriétés de l'objet Transform, voir la classe Transform. Pour plus d'informations sur les propriétés de l'objet Matrix3D, voir la classe Matrix3D.

Déplacement d'un objet dans l'espace 3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez déplacer un objet dans l'espace 3D en modifiant la valeur de ses propriétés `x, y ou z`. Lorsque vous modifiez la valeur de sa propriété `z`, l'objet semble se rapprocher ou s'éloigner de l'observateur.

Le code suivant modifie la valeur de la propriété `z` de deux ellipses en réponse à un événement pour leur imprimer un mouvement de va-et-vient le long de leur axe z. `ellipse2` se déplace plus rapidement qu'`ellipse1` : sa propriété `z` est incrémentée d'un multiple de 20 sur chaque événement Frame, alors que la propriété `z` d'`ellipse1` est incrémentée d'un multiple de 10 :

```
var depth:int = 1000;

function ellipse1FrameHandler(e:Event):void
{
    ellipse1Back = setDepth(e, ellipse1Back);
    e.currentTarget.z += ellipse1Back * 10;
}
function ellipse2FrameHandler(e:Event):void
{
    ellipse2Back = setDepth(e, ellipse1Back);
    e.currentTarget.z += ellipse1Back * 20;
}
function setDepth(e:Event, d:int):int
{
    if(e.currentTarget.z > depth)
    {
        e.currentTarget.z = depth;
        d = -1;
    }
    else if (e.currentTarget.z <  0)
    {
        e.currentTarget.z = 0;
        d = 1;
    }
}
```

Rotation d'un objet dans l'espace 3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez faire pivoter un objet de trois façons différentes, selon la définition de sa propriété de rotation : `rotationX`, `rotationY` et `rotationZ`.

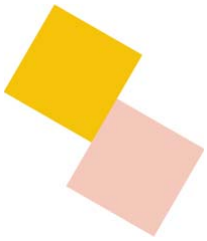
La figure ci-dessous illustre deux carrés qui ne sont pas soumis à une rotation :

Dans la figure suivante, la propriété `rotationY` du conteneur des carrés a été incrémentée pour les faire pivoter sur l'axe y. La rotation du conteneur, ou objet d'affichage parent, des deux carrés fait pivoter ceux-ci :

```
container.rotationY += 10;
```

Dans la figure suivante, la propriété `rotationX` du conteneur des carrés est modifiée : Elle fait pivoter ceux-ci sur l'axe x.

Dans la figure suivante, la propriété `rotationZ` du conteneur des carrés a été incrémentée et ceux-ci pivotent sur l'axe z.



Un objet d'affichage peut se déplacer et pivoter simultanément dans l'espace 3D.

Projection d'objets 3D sur un affichage 2D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La classe PerspectiveProjection du package `flash.geom` facilite l'application d'une perspective rudimentaire lors du déplacement d'objets d'affichage dans l'espace 3D.

Si vous ne créez pas explicitement une projection de perspective pour votre espace 3D, le moteur 3D utilise un objet PerspectiveProjection par défaut, qui existe sur la racine et est propagé à tous ses enfants.

Les trois propriétés qui définissent le mode d'affichage de l'espace 3D par un objet PerspectiveProjection sont les suivantes :

- `fieldOfView`
- `projectionCenter`
- `focalLength`

La modification de la valeur de `fieldOfView` entraîne automatiquement la modification de la valeur de `focalLength`, et inversement, car ces propriétés sont interdépendantes.

La formule permettant de calculer `focalLength` en fonction de la valeur de `fieldOfView` est la suivante :

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

En règle générale, vous modifiez explicitement la propriété `fieldOfView`.

Champ de vision

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

En manipulant la propriété `fieldOfView` de la classe PerspectiveProjection, vous pouvez faire en sorte qu'un objet d'affichage 3D semble s'agrandir ou diminuer selon qu'il se rapproche ou s'éloigne de l'observateur, respectivement.

La propriété `fieldOfView` définit un angle *compris entre* 0 et 180 degrés qui détermine la puissance de la projection de perspective. Plus la valeur est élevée, plus forte est la distorsion appliquée à un objet d'affichage qui se déplace sur son axe z. Une valeur `fieldOfView` basse entraîne une mise à l'échelle minimale et les objets ne semblent reculer que très légèrement. Une valeur élevéeentraîne une plus grande distorsion et une impression plus prononcée de mouvement. La valeur maximale, 179,9999... degrés, produit un effet d'objectif œil-de-poisson extrême. La valeur maximale de `fieldOfView` est 179,9999..., tandis que 0,00001... est sa valeur minimale. Les valeurs exactes de 0 et 180 sont illégales.

Centre de la projection

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `projectionCenter`représente le point de fuite de la projection de perspective. Elle est appliquée comme décalage du point d'alignement par défaut (0,0) dans l'angle supérieur gauche de la scène.

A mesure qu'il semble s'éloigner de l'observateur, un objet s'incline vers le point de fuite et finit par disparaître. Imaginez un couloir extrêmement long. Lorsque vous regardez dans le couloir, les bords des murs convergent vers un point de fuite tout au fond.

Si le point de fuite se trouve au centre de la scène, le couloir disparaît vers un point central. La valeur par défaut de la propriété `projectionCenter` correspond au centre de la scène. Si, par exemple, vous souhaitez que des éléments apparaissent sur la gauche de la scène et une zone 3D sur la droite de la scène, définissez la propriété `projectionCenter` sur un point situé dans la partie droite de la scène pour en faire le point de fuite de votre zone d'affichage 3D.

Distance focale

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `focalLength` représente la distance séparant l'origine du point de vue (0,0,0) de l'emplacement de l'objet d'affichage sur son axe z.

Une distance focale élevée est similaire à un téléobjectif : le champ de vision est étroit et les distances entre les objets compressées. Une distance focale courte s'assimile à un objectif à grand angle et se caractérise par un champ de vision large et une distorsion prononcée. Une distance focale moyenne correspond approximativement à ce que voit l'œil humain.

En règle générale, la propriété `focalLength` est recalculée dynamiquement pendant la transformation de perspective au fur et à mesure du déplacement de l'objet d'affichage, mais il est possible de la définir explicitement.

Valeurs par défaut de la projection de perspective

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'objet PerspectiveProjection par défaut créé sur la racine possède les valeurs suivantes :

- `fieldOfView : 55`
- `perspectiveCenter : stagewidth/2, stageHeight/2`
- `focalLength : stageWidth/ 2 * ( cos(fieldOfView/2) / sin(fieldOfView/2) )`

Ces valeurs sont appliquées si vous ne créez pas votre propre objet PerspectiveProjection.

Vous pouvez instancier votre propre objet PerspectiveProjection dans l'intention de modifier vous-même les propriétés `projectionCenter` et `fieldOfView`. Dans ce cas, les valeurs par défaut du nouvel objet sont les suivantes, en supposant que la scène mesure 500 sur 500 par défaut :

- `fieldOfView : 55`
- `perspectiveCenter : 250,250`
- `focalLength : 480.24554443359375`

Exemple : Projection de perspective

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'exemple suivant illustre l'utilisation de la projection de perspective pour créer l'espace 3D. Il indique comment modifier le point de fuite et la projection de perspective de l'espace par le biais de la propriété `projectionCenter`. Cette modification force un nouveau calcul des propriétés `focalLength` et `fieldOfView`, résultant en une distorsion de l'espace 3D.

Cet exemple :

1. crée un sprite appelé `center`, un cercle avec une mire ;
2. affecte les coordonnées du sprite `center` à la propriété `projectionCenter` de la propriété `perspectiveProjection` de la propriété `transform` de la racine ;
3. ajoute des écouteurs de l'événement souris qui appellent des gestionnaires qui modifient la propriété `projectionCenter` afin qu'elle suive l'emplacement de l'objet `center` ;
4. crée quatre boîtes en accordéon qui forment les murs de l'espace en perspective.

Lorsque vous testez cet exemple, ProjectionDragger.swf, faites glisser le cercle à différents emplacements. Le point de fuite suit le cercle et figure là où vous le déposez. Observez comme les boîtes qui délimitent l'espace s'étirent et se distordent plus vous éloignez le centre de projection du centre de la scène.

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application ProjectionDragger résident dans le dossier Samples/ProjectionDragger.

```
package
{
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.geom.Point;
    import flash.events.*;
    public class ProjectionDragger extends Sprite
    {
        private var center : Sprite;
        private var boxPanel:Shape;
        private var inDrag:Boolean = false;

        public function ProjectionDragger():void
        {
            createBoxes();
            createCenter();
        }
        public function createCenter():void
        {
            var  centerRadius:int = 20;

            center = new Sprite();

            // circle
            center.graphics.lineStyle(1, 0x000099);
            center.graphics.beginFill(0xCCCCCC, 0.5);
            center.graphics.drawCircle(0, 0, centerRadius);
            center.graphics.endFill();
            // cross hairs
            center.graphics.moveTo(0, centerRadius);
            center.graphics.lineTo(0, -centerRadius);
            center.graphics.moveTo(centerRadius, 0);
            center.graphics.lineTo(-centerRadius, 0);
            center.x = 175;
            center.y = 175;
            center.z = 0;
            this.addChild(center);

            center.addEventListener(MouseEvent.MOUSE_DOWN, startDragProjectionCenter);
            center.addEventListener(MouseEvent.MOUSE_UP, stopDragProjectionCenter);
            center.addEventListener( MouseEvent.MOUSE_MOVE, doDragProjectionCenter);
            root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
        }
        public function createBoxes():void
        {
            // createBoxPanel();
            var boxWidth:int = 50;
            var boxHeight:int = 50;
            var numLayers:int = 12;
            var depthPerLayer:int = 50;

            // var boxVec:Vector.<Shape> = new Vector.<Shape>(numLayers);
            for (var i:int = 0; i < numLayers; i++)
            {
                this.addChild(createBox(150, 50, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xCCCCFF));
```

```
                this.addChild(createBox(50, 150, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xFFCCCC));
                this.addChild(createBox(250, 150, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xCCFFCC));
                this.addChild(createBox(150, 250, (numLayers - i) * depthPerLayer, boxWidth,
boxHeight, 0xDDDDDD));
            }
        }

        public function createBox(xPos:int = 0, yPos:int = 0, zPos:int = 100, w:int = 50, h:int
= 50, color:int = 0xDDDDDD):Shape
        {
            var box:Shape = new Shape();
            box.graphics.lineStyle(2, 0x666666);
            box.graphics.beginFill(color, 1.0);
            box.graphics.drawRect(0, 0, w, h);
            box.graphics.endFill();
            box.x = xPos;
            box.y = yPos;
            box.z = zPos;
            return box;
        }
        public function startDragProjectionCenter(e:Event)
        {
            center.startDrag();
            inDrag = true;
        }

        public function doDragProjectionCenter(e:Event)
        {
            if (inDrag)
            {
                root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
            }
        }

        public function stopDragProjectionCenter(e:Event)
        {
            center.stopDrag();
            root.transform.perspectiveProjection.projectionCenter = new Point(center.x,
center.y);
            inDrag = false;
        }
    }
}
```

Pour des projections de perspective plus complexes, utilisez la classe Matrix3D.

Transformations 3D complexes

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La classe Matrix3D vous permet de transformer des points 3D dans un espace de coordonnées ou de mapper des points 3D d'un espace de coordonnées sur un autre.

Il n'est pas nécessaire de comprendre les mathématiques matricielles pour pouvoir utiliser la classe Matrix3D. Ses méthodes permettent de gérer la plupart des opérations de transformation courantes. Il est inutile de vous soucier de la définition explicite ou du calcul de la valeur de chaque élément de la matrice.

Une fois la propriété `z` d'un objet d'affichage définie sur une valeur numérique, vous pouvez récupérer la matrice de transformation de l'objet par le biais de la propriété Matrix3D de l'objet Transform de l'objet d'affichage :

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

Les méthodes de l'objet Matrix3D vous permettent d'opérer une translation sur un objet d'affichage, de le faire pivoter et de le mettre à l'échelle, ainsi que de lui appliquer une projection de perspective.

Utilisez la classe Vector3D et ses propriétés `x`, `y` et `z` pour gérer les points 3D. Elle peut également représenter en physique un vecteur spatial, doté d'une direction et d'une magnitude. Les méthodes de la classe Vector3D vous permettent d'effectuer des calculs courants portant sur des vecteurs spatiaux : somme, produit scalaire et produit vectoriel, par exemple.

Remarque : *la classe Vector3D n'a aucun rapport avec la classe Vector d'ActionScript. La classe Vector3D contient des propriétés et des méthodes permettant de définir et de manipuler les points 3D, alors que la classe Vector prend en charge les tableaux d'objets typés.*

Création d'objets Matrix3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez créer ou récupérer des objets `Matrix3D` de trois façons principales :

- Utilisez la méthode constructeur `Matrix3D()` pour instancier une nouvelle matrice. Le constructeur `Matrix3D()` gère un objet `Vector` contenant 16 valeurs numériques et place chacune d'elles dans une cellule de la matrice. Exemple :

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- Définissez la valeur de la propriété `z` d'un objet d'affichage. Récupérez ensuite la matrice de transformation de la propriété `transform.matrix3D` de cet objet.
- Récupérez l'objet Matrix3D qui régit l'affichage des objets 3D sur la scène en appelant la méthode `perspectiveProjection.toMatrix3D()` sur l'objet d'affichage racine.

Application de plusieurs transformations 3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez appliquer simultanément de nombreuses transformations 3D à l'aide d'un objet Matrix3D. Ainsi, pour faire pivoter, mettre à l'échelle, puis déplacer un cube, vous pourriez appliquer trois transformations distinctes à chacun de ses points. Il est cependant beaucoup plus efficace de précalculer plusieurs transformations dans un même objet Matrix3D et d'appliquer une transformation matricielle unique à chacun des points.

Remarque : *l'ordre d'application des transformations matricielles est important. Les calculs matriciels ne sont pas réversibles. Ainsi, l'application d'une rotation puis d'une translation ne donne pas le même résultat que l'opération inverse.*

L'exemple suivant illustre deux façons d'appliquer plusieurs transformations 3D.

```
package {
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;
    import flash.geom.*;

public class Matrix3DTransformsExample extends Sprite
    {
        private var rect1:Shape;
        private var rect2:Shape;

public function Matrix3DTransformsExample():void
        {
            var pp:PerspectiveProjection = this.transform.perspectiveProjection;
            pp.projectionCenter = new Point(275,200);
            this.transform.perspectiveProjection = pp;

            rect1 = new Shape();
            rect1.x = -70;
            rect1.y = -40;
            rect1.z = 0;
            rect1.graphics.beginFill(0xFF8800);
            rect1.graphics.drawRect(0,0,50,80);
            rect1.graphics.endFill();
            addChild(rect1);

            rect2 = new Shape();
            rect2.x = 20;
            rect2.y = -40;
            rect2.z = 0;
            rect2.graphics.beginFill(0xFF0088);
            rect2.graphics.drawRect(0,0,50,80);
            rect2.graphics.endFill();
            addChild(rect2);
```

```
            doTransforms();
        }

        private function doTransforms():void
        {
            rect1.rotationX = 15;
            rect1.scaleX = 1.2;
            rect1.x += 100;
            rect1.y += 50;
            rect1.rotationZ = 10;

            var matrix:Matrix3D = rect2.transform.matrix3D;
            matrix.appendRotation(15, Vector3D.X_AXIS);
            matrix.appendScale(1.2, 1, 1);
            matrix.appendTranslation(100, 50, 0);
            matrix.appendRotation(10, Vector3D.Z_AXIS);
            rect2.transform.matrix3D = matrix;
        }
    }
}
```

Dans la méthode `doTransforms()`, le premier bloc de code utilise les propriétés DisplayObject pour modifier la rotation, la mise à l'échelle et la position d'un rectangle. Le second bloc utilise les méthodes de la classe Matrix3D pour effectuer des transformations identiques.

L'utilisation des méthodes `Matrix3D` présente un avantage principal : tous les calculs sont déjà effectués dans la matrice. Ils sont ensuite appliqués une seule fois à l'objet d'affichage, lors de la définition de sa propriété `transform.matrix3D`. La définition des propriétés DisplayObject améliore quelque peu la lisibilité du code source. Cependant, chaque définition d'une propriété de rotation ou de mise à l'échelle donne lieu à plusieurs calculs et entraîne la modification de plusieurs propriétés de l'objet d'affichage.

Si votre code applique plusieurs fois des transformations complexes identiques à des objets d'affichage, enregistrez l'objet Matrix3D en tant que variable, puis réappliquez-le autant de fois que nécessaire.

Réorganisation de l'affichage à l'aide d'objets Matrix3D

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Comme indiqué plus haut, l'ordre d'apparition des objets d'affichage dans la liste d'affichage détermine l'ordre d'apparition à l'affichage, quels que soient leurs axes z relatifs. Si votre animation transforme les propriétés d'objets d'affichage dans un ordre différent de celui de la liste d'affichage, l'ordre d'apparition des objets d'affichage ne correspondra peut-être pas à celui des axes z. Un objet qui devrait sembler plus éloigné de l'observateur risque donc d'apparaître devant un objet plus proche.

Pour avoir la certitude que l'ordre d'apparition des objets d'affichage 3D correspond à leur profondeur relative, procédez comme suit :

1. A l'aide de la méthode `getRelativeMatrix3D()` de l'objet Transform, extrayez la profondeur relative (`z-axes`) des objets d'affichage 3D enfant.
2. Supprimez les objets de la liste d'affichage à l'aide de la méthode `removeChild()`.
3. Triez les objets d'affichage en fonction de leurs valeurs d'axe z relatives.
4. Ajoutez à nouveau les enfants à la liste d'affichage en ordre inverse, par le biais de la méthode `addChild()`.

Cette réorganisation garantit que vos objets s'affichent conformément à leurs axes z relatifs.

Le code suivant garantit l'affichage correct d'une boîte 3D à six faces. Il réorganise les faces de la boîte une fois que des rotations lui ont été appliquées.

```
public var faces:Array; . . .

public function ReorderChildren()
{
    for(var ind:uint = 0; ind < 6; ind++)
    {
        faces[ind].z = faces[ind].child.transform.getRelativeMatrix3D(root).position.z;
        this.removeChild(faces[ind].child);
    }
    faces.sortOn("z", Array.NUMERIC | Array.DESCENDING);
    for (ind = 0; ind < 6; ind++)
    {
        this.addChild(faces[ind].child);
    }
}
```

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application résident dans le dossier Samples/ReorderByZ.

Création d'effets 3D à l'aide de triangles

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Dans ActionScript, vous effectuez des transformations de bitmap à l'aide de la méthode `Graphics.drawTriangles()`, car les modèles 3D sont représentés par un ensemble de triangles dans l'espace. (Flash Player et AIR ne prennent néanmoins pas en charge une mémoire tampon de profondeur, car les objets d'affichage sont toujours essentiellement plats, ou 2D. Pour plus d'informations, voir « Présentation des objets d'affichage 3D dans les moteurs d'exécution de Flash Player et d'AIR » à la page 363.) La méthode `Graphics.drawTriangles()` est similaire à la méthode `Graphics.drawPath()`, en ce qu'elle accepte un ensemble de coordonnées pour tracer un tracé triangulaire.

Pour vous familiariser avec l'utilisation de `Graphics.drawPath()`, voir « Tracés de dessin » à la page 244.

La méthode `Graphics.drawTriangles()` utilise une propriété Vector.<Number> pour spécifier l'emplacement des points sur le tracé triangulaire :

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number>
= null, culling:String = "none"):void
```

Le premier paramètre de `drawTriangles()`, `vertices`, est le seul paramètre obligatoire. C'est un vecteur de nombres définissant les coordonnées par lesquelles vos triangles sont tracés. Trois ensembles de coordonnées (six nombres) représentent un tracé triangulaire. Sans le paramètre `indices`, la longueur du vecteur doit systématiquement être un facteur de six, car chaque triangle nécessite trois paires de coordonnées (trois ensembles de deux valeurs x/y). Exemple :

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

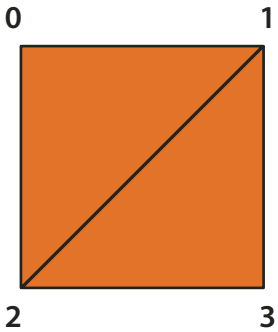
Ces triangles n'ont pas de points en commun, mais si tel était le cas, le second paramètre de `drawTriangles()`, `indices`, permettrait de réutiliser des valeurs du vecteur `vertices` pour plusieurs triangles.

Lorsque vous utilisez le paramètre `indices`, gardez à l'esprit le fait que les valeurs `indices` représentent des index de point, pas des index en rapport direct avec les éléments du tableau `vertices`. En d'autres termes, un index du vecteur `vertices` tel qu'il est défini par `indices` correspond en fait à l'index réel divisé par 2. Pour le troisième point d'un vecteur `vertices`, par exemple, utilisez une valeur `indices` de 2, même si la première valeur numérique de ce point commence à l'index vectoriel 4.

Par exemple, fusionnez deux triangles de sorte qu'ils aient en commun le bord diagonal, par le biais du paramètre `indices` :

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

Vous remarquerez que, bien qu'un carré résulte du tracé de deux triangles, seuls quatre points ont été spécifiés dans le vecteur `vertices`. Grâce à `indices`, les deux points partagés par les deux triangles sont réutilisés pour chacun d'eux. Le nombre total de sommets passe donc de 6 (12 nombres) à 4 (8 nombres).



Carré tracé à l'aide de deux triangles à l'aide du paramètre vertices

Cette technique s'avère utile pour les maillages triangulaires plus grands, dans lesquels la plupart des points sont partagés par plusieurs triangles.

Il est possible d'appliquer tous les remplissages aux triangles. Ils sont appliqués au maillage triangulaire résultant comme ils le seraient à toute autre forme.

Transformation d'images bitmap

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les transformations de bitmap donnent une illusion de perspective ou « texture » à un objet en trois dimensions. Vous pouvez notamment distordre une bitmap en direction d'un point de fuite, afin que l'image semble diminuer à mesure qu'elle se rapproche de celui-ci. Vous pouvez aussi utiliser une bitmap en deux dimensions pour créer une surface sur un objet en trois dimensions, donnant ainsi l'impression qu'il possède une texture ou « enveloppe ».

Surface en deux dimensions utilisant un point de fuite et objet en trois dimensions enveloppé dans une bitmap

Mappage des coordonnées UV

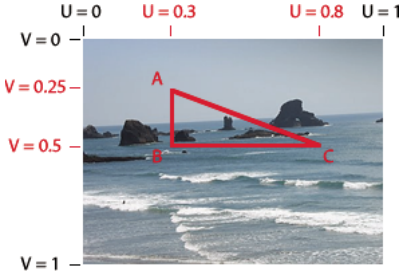
Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Lorsque vous commencerez à manipuler les textures, vous souhaiterez utiliser le paramètre uvtData de `drawTriangles()`. Ce paramètre vous permet de définir le mappage des coordonnées UV pour les remplissages de bitmap.

Le mappage des coordonnées UV est une méthode d'application d'une texture à des objets. Il repose sur deux valeurs, une valeur U horizontale (x) et une valeur V verticale (y). Ces valeurs sont basées sur des pourcentages et non sur des valeurs de pixels. 0 U et 0 V correspondent au coin supérieur gauche d'une image, 1 U et 1 V à son coin inférieur droit :

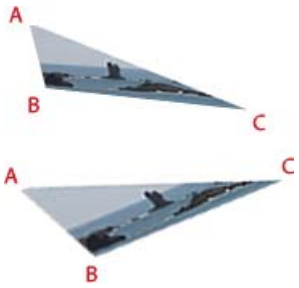
Emplacements UV 0 et 1 sur une image bitmap

Il est possible d'affecter des coordonnées UV aux vecteurs d'un triangle de sorte qu'ils s'associent aux emplacements respectifs sur une image :



Coordonnées UV d'une zone triangulaire sur une image bitmap

Les valeurs UV restent en phase avec les points du triangle.



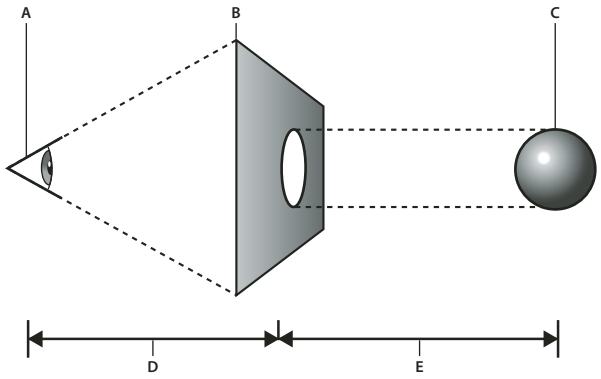
Les sommets du triangle se déplacent et l'image bitmap se distord pour que les valeurs UV d'un point individuel restent identiques.

Au fur et à mesure que des transformations 3D ActionScript sont appliquées au triangle associé à la bitmap, celle-ci est appliquée au triangle en fonction des valeurs UV. Par conséquent, au lieu d'utiliser des calculs matriciels, définissez ou réglez les valeurs UV pour créer un effet tridimensionnel.

La méthode `Graphics.drawTriangles()` accepte également une information facultative pour les transformations tridimensionnelles : la valeur T. La valeur T de uvtData représente la perspective 3D ou, plus spécifiquement, le facteur d'échelle du sommet associé. Le mappage des coordonnées UVT ajoute une correction de perspective au mappage des coordonnées UV. Par exemple, si un objet de l'espace 3D est éloigné du point de vue de telle sorte qu'il semble mesurer 50 % de sa taille « d'origine », sa valeur T correspond à 0,5. Comme les objets de l'espace 3D sont représentés à l'aide de triangles, l'emplacement de ceux-ci le long de l'axe z détermine leur valeur T. L'équation qui représente la valeur T est la suivante :

```
T = focalLength/(focalLength + z);
```

Dans cette équation, focalLength représente une distance focale ou un emplacement à l'« écran » calculé qui détermine la quantité de perspective de l'affichage.



Distance focale et valeur z
***A.** point de vue **B.** écran **C.** objet 3D **D.** valeur focalLength **E.** valeur z*

La valeur T permet de mettre à l'échelle des formes simples pour donner l'impression qu'elles se trouvent au loin. C'est généralement la valeur utilisée pour convertir les points 3D en points 2D. Dans le cas des données UVT, elle permet aussi de mettre à l'échelle une bitmap entre les points d'un triangle avec perspective.

Lorsque vous définissez des valeurs UVT, la valeur T suit directement les valeurs UV définies pour un sommet. Avec l'inclusion de T, chaque trio de valeurs du paramètre `uvtData` (U, V et T) correspond à chaque paire de valeurs du paramètre `vertices` (x et y). Avec les valeurs UV seules, uvtData.length == vertices.length. Avec l'inclusion d'une valeur T, uvtData.length = 1,5*vertices.length.

L'exemple suivant illustre un plan qui pivote, par le biais de données UVT, dans un espace 3D. Il utilise l'image ocean.jpg et une classe « d'interaction », ImageLoader, qui charge l'image afin qu'il soit possible de l'affecter à l'objet BitmapData.

La source de la classe ImageLoader (enregistrez ce code dans le fichier ImageLoader.as) se présente comme suit :

```
package {
    import flash.display.*
    import flash.events.*;
    import flash.net.URLRequest;
    public class ImageLoader extends Sprite {
        public var url:String;
        public var bitmap:Bitmap;
    public function ImageLoader(loc:String = null) {
        if (loc != null){
            url = loc;
            loadImage();
        }
    }
    public function loadImage():void{
        if (url != null){
            var loader:Loader = new Loader();
            loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);
            loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, onIoError);

                var req:URLRequest = new URLRequest(url);
                loader.load(req);
            }
        }

    private function onComplete(event:Event):void {
            var loader:Loader = Loader(event.target.loader);
            var info:LoaderInfo = LoaderInfo(loader.contentLoaderInfo);
            this.bitmap = info.content as Bitmap;
            this.dispatchEvent(new Event(Event.COMPLETE));
    }

    private function onIoError(event:IOErrorEvent):void {
            trace("onIoError: " + event);
    }
    }
}
```

Le code ActionScript qui utilise des triangles, le mappage des coordonnées UV et des valeurs T pour que l'image semble pivoter et diminuer au fur et à mesure qu'elle se rapproche d'un point de fuite est indiqué ci-dessous. Enregistrez-le dans un fichier que vous nommerez Spinning3dOcean.as :

```
package {
    import flash.display.*
    import flash.events.*;
    import flash.utils.getTimer;

    public class Spinning3dOcean extends Sprite {
        // plane vertex coordinates (and t values)
        var x1:Number = -100,y1:Number = -100,z1:Number = 0,t1:Number = 0;
        var x2:Number = 100,y2:Number = -100,z2:Number = 0,t2:Number = 0;
        var x3:Number = 100,y3:Number = 100,z3:Number = 0,t3:Number = 0;
        var x4:Number = -100,y4:Number = 100,z4:Number = 0,t4:Number = 0;
        var focalLength:Number = 200;
        // 2 triangles for 1 plane, indices will always be the same
        var indices:Vector.<int>;

        var container:Sprite;

        var bitmapData:BitmapData; // texture
        var imageLoader:ImageLoader;
        public function Spinning3dOcean():void {
            indices =  new Vector.<int>();
            indices.push(0,1,3, 1,2,3);

            container = new Sprite(); // container to draw triangles in
            container.x = 200;
            container.y = 200;
            addChild(container);

            imageLoader = new ImageLoader("ocean.jpg");
            imageLoader.addEventListener(Event.COMPLETE, onImageLoaded);
        }
        function onImageLoaded(event:Event):void {
            bitmapData = imageLoader.bitmap.bitmapData;
            // animate every frame
            addEventListener(Event.ENTER_FRAME, rotatePlane);
        }
        function rotatePlane(event:Event):void {
            // rotate vertices over time
            var ticker = getTimer()/400;
            z2 = z3 = -(z1 = z4 = 100*Math.sin(ticker));
            x2 = x3 = -(x1 = x4 = 100*Math.cos(ticker));

            // calculate t values
```

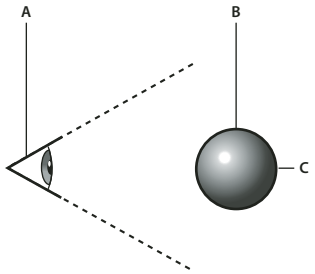
```
            t1 = focalLength/(focalLength + z1);
            t2 = focalLength/(focalLength + z2);
            t3 = focalLength/(focalLength + z3);
            t4 = focalLength/(focalLength + z4);

            // determine triangle vertices based on t values
            var vertices:Vector.<Number> = new Vector.<Number>();
            vertices.push(x1*t1,y1*t1, x2*t2,y2*t2, x3*t3,y3*t3, x4*t4,y4*t4);
            // set T values allowing perspective to change
            // as each vertex moves around in z space
            var uvtData:Vector.<Number> = new Vector.<Number>();
            uvtData.push(0,0,t1, 1,0,t2, 1,1,t3, 0,1,t4);

            // draw
            container.graphics.clear();
            container.graphics.beginBitmapFill(bitmapData);
            container.graphics.drawTriangles(vertices, indices, uvtData);
        }
    }
}
```

Pour tester cet exemple, enregistrez ces deux fichiers de classe dans le même répertoire qu'une image nommée « ocean.jpg ». Vous pouvez constater la transformation appliquée à la bitmap d'origine pour qu'elle semble disparaître au loin et pivoter dans l'espace 3D.

Culling

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le culling est un processus qui détermine quelles surfaces d'un objet en trois dimensions ne doivent pas être rendues par le moteur de rendu car elles ne sont pas visibles à partir du point de vue actuel. Dans l'espace 3D, la surface « arrière » d'un objet en trois dimensions n'est pas visible à partir du point de vue.



L'arrière d'un objet 3D n'est pas visible à partir du point de vue.
***A.** point de vue **B.** objet 3D **C.** arrière d'un objet en trois dimensions*

Par essence, tous les triangles sont systématiquement rendus, quelles que soient leur taille, forme et position. Le culling (élimination) garantit que Flash Player ou AIR rend votre objet 3D correctement. En outre, pour éviter les cycles de rendu superflus, il est parfois souhaitable que le moteur de rendu omette certains triangles. Soit un cube en rotation dans l'espace. A tout moment donné, vous ne voyez jamais plus de trois de ses faces car celles qui ne sont pas visibles font face à l'autre direction, de l'autre côté du cube. Comme ces faces ne sont pas visibles, il est inutile que le moteur de rendu les trace. Sans élimination (culling), Flash Player ou AIR rend les faces avant et arrière.

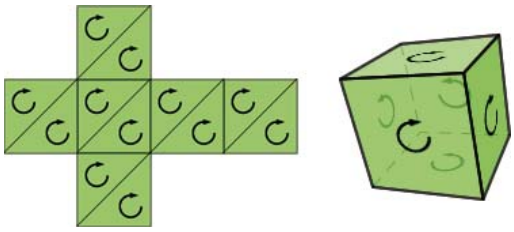
Certaines faces d'un cube ne sont pas visibles à partir du point de vue actuel.

C'est pourquoi la méthode `Graphics.drawTriangles()` gère un quatrième paramètre qui permet de définir une valeur de culling :

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

Le paramètre culling correspond à une valeur de la classe d'énumération `TriangleCulling` : `TriangleCulling.NONE`, `TriangleCulling.POSITIVE` et `TriangleCulling.NEGATIVE`. Ces valeurs sont fonction de la direction du tracé triangulaire définissant la surface de l'objet. L'API ActionScript qui permet de déterminer le culling considère comme acquis que tous les triangles orientés vers l'extérieur d'une forme 3D sont tracés dans le même sens. Le retournement d'une face de triangle entraîne un changement de direction du tracé. A ce point, il est possible de supprimer le triangle du rendu.

Si la valeur de `TriangleCulling` est définie sur `POSITIVE`, les triangles dont le tracé a une direction positive (sens horaire) sont supprimés. Si la valeur de `TriangleCulling` est définie sur `NEGATIVE`, les triangles dont le tracé a une direction négative (sens antihoraire) sont supprimés. Dans le cas d'un cube, les surfaces orientées vers l'avant ont un tracé à la direction positive et les surfaces orientées vers l'arrière, un tracé à la direction négative.



Cube « déroulé » illustrant le sens du tracé. Lorsque le cube est « enroulé », le sens du tracé de la face arrière est inversé.

Pour comprendre le fonctionnement du processus d'élimination (culling), reprenez l'exemple de la section « Mappage des coordonnées UV » à la page 377 et définissez le paramètre de culling de la méthode `drawTriangles()` sur `TriangleCulling.NEGATIVE` :

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

Vous pouvez constater que la face « arrière » de l'image n'est pas rendue lorsque l'objet pivote.

Chapitre 20 : Principes de base de l'utilisation du texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour afficher du texte à l'écran dans Adobe® Flash® Player ou Adobe® AIR™, utilisez une occurrence de la classe TextField ou les classes Text Engine de Flash. Ces classes vous permettent de créer, d'afficher et de mettre en forme du texte. Vous pouvez également utiliser Text Layout Framework (TLF). Il s'agit d'une bibliothèque de composants basée sur les classes Flash Text Engine, mais conçue de manière à offrir une utilisation conviviale. Sur les périphériques mobiles, vous pouvez utiliser la classe StageText pour saisir du texte.

Vous pouvez établir le contenu spécifique de champs de texte ou désigner la source du texte, puis en définir l'aspect. Vous pouvez également répondre aux événements utilisateur (saisie de texte ou clic sur un lien hypertexte).

Les classes TextField et Flash Text Engine vous permettent d'afficher et de gérer le texte dans Flash Player et AIR. Vous disposez de la classe TextField pour créer des objets texte à des fins d'affichage et d'entrée. La classe TextField sert de base à d'autres composants texte tels que TextArea et TextInput. La classe TextFormat permet de définir la mise en forme de caractère et paragraphe des objets TextField et vous pouvez appliquer des feuilles de style en cascade (CSS) à l'aide de la propriété Textfield.styleSheet et de la classe StyleSheet. Vous pouvez affecter directement à un champ de texte un texte au format HTML, qui peut contenir des médias intégrés (clips, fichiers SWF, fichiers GIF, fichiers PNG et fichiers JPEG).

Flash Text Engine (FTE), disponible à partir de Flash Player 10 et Adobe AIR 1.5, propose une prise en charge de bas niveau pour un contrôle sophistiqué des mesures de texte, de la mise en forme et du texte bidirectionnel. Il se caractérise également par un flux de texte optimisé et une prise en charge des langues enrichie. Bien que vous puissiez utiliser Flash Text Engine pour créer et gérer des éléments texte, il est essentiellement destiné à générer des composants de manipulation du texte et nécessite des compétences accrues en matière de programmation. Text Layout Framework, qui comprend un composant de manipulation du texte basé sur Flash Text Engine, facilite l'utilisation des fonctions avancées du nouveau moteur texte. TLF est une bibliothèque extensible reposant entièrement sur ActionScript 3.0. Vous pouvez utiliser le composant TLF existant ou utiliser la structure pour créer votre propre composant de texte.

La classe StageText, disponible à partir d'AIR 3, fournit un champ de saisie de texte natif. Etant donné que ce champ est mis à disposition par le système d'exploitation du périphérique, il fournit l'expérience avec laquelle les utilisateurs d'un périphérique sont le plus familiarisés. Une occurrence de StageText n'est pas un objet d'affichage. Plutôt que de l'ajouter à la liste d'affichage, affectez une scène à une occurrence, ainsi qu'une zone d'affichage sur cette scène appelée fenêtre d'affichage. L'occurrence de StageText s'affiche face à tous les objets d'affichage.

Pour plus d'informations, voir :

- « Utilisation de la classe TextField » à la page 385
- « Utilisation de Flash Text Engine » à la page 410
- « Utilisation de Text Layout Framework » à la page 440
- Native text input with StageText (disponible en anglais uniquement)

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la manipulation du texte :

Feuilles de style en cascade Syntaxe standardisée permettant de définir les styles et la mise en forme du texte structuré en XML et en HTML.

Police de périphérique Police installée sur l'ordinateur de l'utilisateur.

Champ de texte dynamique Champ de texte dont le contenu peut être modifié en ActionScript, mais pas par l'utilisateur.

Police incorporée Police de caractères dont les données, sous forme vectorielle, sont enregistrées dans le fichier SWF de l'application.

Texte HTML Texte inséré dans un champ de texte en ActionScript qui, outre le contenu à proprement parler, comporte des balises HTML de mise en forme.

Champ de saisie de texte Champ de texte dont le contenu peut être modifié soit en ActionScript, soit par l'utilisateur.

Crénage Réglage de l'espace entre les paires de caractères de sorte à uniformiser l'espacement des mots et à améliorer la lisibilité du texte.

Champ de texte statique Champ de texte créé dans l'environnement de création, dont le contenu ne peut pas être modifié pendant l'exécution du fichier SWF.

Métrique des lignes de texte Mesure de la taille des diverses parties du texte figurant dans un champ de texte : ligne de base du texte, hauteur du sommet des caractères, taille des jambages (partie de certaines minuscules qui s'étend sous la ligne de base), etc.

Interlettrage Réglage de l'espacement entre des groupes de lettres ou des blocs de texte en vue d'augmenter ou de réduire la densité pour améliorer la lisibilité du texte.

Chapitre 21 : Utilisation de la classe TextField

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser une occurrence de la classe TextField pour afficher du texte ou créer un champ de saisie de texte à l'écran dans Adobe® Flash® Player ou Adobe® AIR™. La classe TextField sert de base à d'autres composants de texte tels que TextArea ou TextInput.

Le contenu des champs de texte peut être spécifié à l'avance dans le fichier SWF, chargé à partir d'un fichier texte ou d'une base de données, ou saisi par l'utilisateur dans votre application. Au sein du champ lui-même, le texte peut être du contenu HTML, avec des images incorporées. Après avoir créé une occurrence de champ de texte, vous pouvez utiliser les classes flash.text, telles que TextFormat et StyleSheet, pour contrôler l'aspect du texte. Le package flash.text contient la quasi-totalité des classes relatives à la création, à la gestion et au formatage de texte dans ActionScript.

Pour mettre en forme du texte, il est nécessaire de créer un objet TextFormat et de l'affecter au champ de texte. Si le champ de texte contient du texte en HTML, vous pouvez lui appliquer un objet StyleSheet pour affecter des styles à des éléments spécifiques du texte. L'objet TextFormat ou StyleSheet contient des propriétés qui définissent l'aspect du texte, par exemple sa couleur, sa taille et sa graisse. L'objet TextFormat attribue des propriétés à l'ensemble du contenu d'un champ de texte, ou à une partie du texte seulement. Par exemple, au sein du même champ de texte, une phrase peut être en gras et en rouge, puis la suivante en italique et en bleu.

Outre les classes du package flash.text, la classe flash.events.TextEvent permet de répondre aux actions de l'utilisateur liées au texte.

Voir aussi

« Attribution de formats texte » à la page 393

« Affichage du texte HTML » à la page 387

« Application de feuilles de style en cascade » à la page 394

Affichage du texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Bien que les outils de programmation tels qu'Adobe Flash Builder et Flash Professional offrent plusieurs options d'affichage du texte (composants liés au texte ou outils texte), la méthode d'affichage de texte par programmation la plus simple consiste à utiliser un champ de texte.

Types de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le type de texte d'un champ de texte est caractérisé par sa source :

- Texte dynamique

  Le texte dynamique correspond au contenu chargé à partir d'une source externe, telles qu'un fichier texte ou xml, ou un service Web.

- Saisie de texte

  Le texte saisi est le texte entré par l'utilisateur ou du texte dynamique que l'utilisateur peut modifier. Vous pouvez définir une feuille de style pour formater le texte saisi, ou utiliser la classe flash.text.TextFormat pour attribuer au champ de texte des propriétés destinées au texte saisi. Pour plus d'informations, voir « Capture du texte saisi par l'utilisateur » à la page 391.

- Texte statique

  Le texte statique est créé par le biais de Flash Professional uniquement. Vous ne pouvez pas créer une occurrence de texte à l'aide d'ActionScript 3.0. Vous pouvez néanmoins utiliser les classes ActionScript telles que StaticText et TextSnapshot pour manipuler une occurrence de texte statique existante. Pour plus d'informations, voir « Utilisation du texte statique » à la page 399.

Modification du contenu d'un champ de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez définir du texte dynamique en affectant une chaîne à la propriété `flash.text.TextField.text`. La chaîne est directement affectée à la propriété, comme suit :

```
myTextField.text = "Hello World";
```

Vous pouvez également affecter à la propriété `text` une valeur issue d'une variable définie dans votre code, comme dans l'exemple suivant :

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class TextWithImage extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello World";

        public function TextWithImage()
        {
            addChild(myTextBox);
            myTextBox.text = myText;
        }
    }
}
```

Vous pouvez également attribuer à la propriété `text` une valeur issue d'une variable distante. Le chargement de valeurs textuelles à partir de sources distantes peut se faire de trois manières :

- Les classes flash.net.URLLoader et flash.net.URLRequest chargent des variables à partir d'emplacements locaux ou distants.
- L'attribut `FlashVars` est incorporé dans la page HTML qui héberge le fichier SWF et peut contenir des valeurs destinées aux variables de texte.
- La classe flash.net.SharedObject gère le stockage persistant des valeurs. Pour plus d'informations, voir « Stockage des données locales » à la page 728.

Affichage du texte HTML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `htmlText` de la classe flash.text.TextField permet d'indiquer que la chaîne de texte contient des balises HTML de formatage du contenu. Comme le montre l'exemple suivant, vous devez affecter votre chaîne à la propriété `htmlText` (et non à la propriété `text`) pour que Flash Player ou AIR puisse afficher le texte sous forme HTML :

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as <u>HTML</u> text.</p>";
myTextBox.htmlText = myText;
```

Pour la propriété `htmlText`, Flash Player et AIR prennent en charge un sous-ensemble de balises et d'entités HTML. La description de propriété `flash.text.TextField.htmlText` dans le manuel Guide de référence ActionScript 3.0 pour Flash Professional fournit des informations détaillées sur les balises et entités HTML prises en charge.

Une fois que vous avez spécifié votre contenu à l'aide de la propriété `htmlText`, vous pouvez utiliser des feuilles de style ou la balise `textformat` pour gérer le formatage. Pour plus d'informations, voir « Mise en forme du texte » à la page 393.

Utilisation d'images dans des champs de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'affichage du contenu sous forme de texte HTML présente un autre avantage : vous pouvez inclure des images dans le champ de texte. Il est possible de référencer une image, locale ou distante, grâce à la balise `img` et de la faire apparaître dans le champ de texte associé.

L'exemple suivant crée un champ de texte appelé `myTextBox` et incorpore au texte une image JPG représentant un œil, image stockée dans le même répertoire que le fichier SWF :

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class TextWithImage extends Sprite
    {
        private var myTextBox:TextField;
        private var myText:String = "<p>This is <b>some</b> content to <i>test</i> and
<i>see</i></p><p><img src='eye.jpg' width='20' height='20'></p><p>what can be
rendered.</p><p>You should see an eye image and some <u>HTML</u> text.</p>";

        public function TextWithImage()
        {
            myTextBox.width = 200;
            myTextBox.height = 200;
            myTextBox.multiline = true;
            myTextBox.wordWrap = true;
            myTextBox.border = true;

            addChild(myTextBox);
            myTextBox.htmlText = myText;
        }
    }
}
```

La balise `img` prend en charge les fichiers JPEG, GIF, PNG et SWF.

Défilement du texte dans un champ de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans bien des cas, votre texte peut s'avérer plus long que le champ de texte qui le contient. Il se peut également qu'un champ de saisie permette à l'utilisateur de saisir plus de caractères qu'il ne peut en afficher en une seule fois. Les propriétés de défilement de la classe flash.text.TextField permettent de gérer du contenu long, que ce soit verticalement ou horizontalement.

Ces propriétés sont les suivantes : `TextField.scrollV`, `TextField.scrollH`, `maxScrollV` et `maxScrollH`. Utilisez-les pour répondre à des événements tels qu'un clic de souris ou une pression sur une touche.

L'exemple ci-après crée un champ de texte de taille fixe et contenant plus de texte que le champ ne peut afficher en une seule fois. Lorsque l'utilisateur clique sur le champ de texte, le texte défile verticalement.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;
    import flash.events.MouseEvent;

    public class TextScrollExample extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello world and welcome to the show. It's really nice to
meet you. Take your coat off and stay a while. OK, show is over. Hope you had fun. You can go
home now. Don't forget to tip your waiter. There are mints in the bowl by the door. Thank you.
Please come again.";

        public function TextScrollExample()
        {
            myTextBox.text = myText;
            myTextBox.width = 200;
            myTextBox.height = 50;
            myTextBox.multiline = true;
            myTextBox.wordWrap = true;
            myTextBox.background = true;
            myTextBox.border = true;

            var format:TextFormat = new TextFormat();
            format.font = "Verdana";
            format.color = 0xFF0000;
            format.size = 10;

            myTextBox.defaultTextFormat = format;
            addChild(myTextBox);
            myTextBox.addEventListener(MouseEvent.MOUSE_DOWN, mouseDownScroll);
        }

        public function mouseDownScroll(event:MouseEvent):void
        {
            myTextBox.scrollV++;
        }
    }
}
```

Sélection et manipulation de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez sélectionner du texte, qu'il soit dynamique ou saisi. Les propriétés et méthodes de sélection de texte de la classe TextField utilisent des positions d'index pour déterminer l'étendue du texte à manipuler. Vous pouvez donc programmer la sélection du texte saisi ou dynamique, même si vous n'en connaissez pas le contenu.

Remarque : *si vous choisissez l'option sélectionnable associée à un champ de texte statique dans Flash Professional, le champ de texte exporté et placé dans la liste d'affichage est un champ de texte dynamique normal.*

Sélection du texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `flash.text.TextField.selectable` a la valeur `true` par défaut. Vous pouvez en outre sélectionner du texte par code à l'aide de la méthode `setSelection()`.

Par exemple, pour sélectionner un texte spécifique dans un champ de texte lorsque l'utilisateur clique dans ce dernier :

```
var myTextField:TextField = new TextField();
myTextField.text = "No matter where you click on this text field the TEXT IN ALL CAPS is selected.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.CLICK, selectText);

function selectText(event:MouseEvent):void
{
    myTextField.setSelection(49, 65);
}
```

De même, pour que le texte d'un champ de texte soit sélectionné dès son affichage initial, créez une fonction de gestion d'événement qui sera appelée lorsque le champ de texte sera ajouté à la liste d'affichage.

Capture du texte sélectionné par l'utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les propriétés `selectionBeginIndex` et `selectionEndIndex` de TextField, qui sont en lecture seule (et ne peuvent donc pas être utilisées à l'aide de code pour sélectionner du texte), permettent également de capturer la sélection actuelle effectuée par l'utilisateur. Par ailleurs, les champs de texte saisis peuvent utiliser la propriété `caretIndex`.

Par exemple, ce code renvoie les valeurs d'index du texte sélectionné par l'utilisateur :

```
var myTextField:TextField = new TextField();
myTextField.text = "Please select the TEXT IN ALL CAPS to see the index values for the first
and last letters.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.MOUSE_UP, selectText);

function selectText(event:MouseEvent):void
{
    trace("First letter index position: " + myTextField.selectionBeginIndex);
    trace("Last letter index position: " + myTextField.selectionEndIndex);
}
```

Vous pouvez également appliquer un ensemble de propriétés de l'objet TextFormat à la sélection pour modifier l'aspect du texte. Pour plus d'informations sur l'application d'un ensemble de propriétés TextFormat au texte sélectionné, voir « Formatage de plages de texte au sein d'un champ de texte » à la page 396.

Capture du texte saisi par l'utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, la propriété `type` d'un champ de texte est définie sur `dynamic`. Si vous attribuez à cette propriété `type` la valeur `input` à l'aide de la classe TextFieldType, vous pouvez recueillir la saisie de l'utilisateur et enregistrer cette valeur pour l'utiliser dans d'autres zones de l'application. Les champs de texte saisi sont utiles dans les formulaires et toute autre application qui attend que l'utilisateur définisse une valeur de texte à utiliser ailleurs dans le programme.

Par exemple, le code suivant crée un champ de texte de saisie appelé `myTextBox`. Lorsque l'utilisateur saisit du texte dans le champ, l'événement `textInput` est déclenché. Un gestionnaire d'événement appelé `textInputCapture` capture la chaîne de texte saisie et l'attribue à une variable. Flash Player ou AIR affiche le nouveau texte dans un autre champ de texte appelé `myOutputBox`.

```
package
{
    import flash.display.Sprite;
    import flash.display.Stage;
    import flash.text.*;
    import flash.events.*;

    public class CaptureUserInput extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myOutputBox:TextField = new TextField();
        private var myText:String = "Type your text here.";

        public function CaptureUserInput()
        {
            captureText();
        }

        public function captureText():void
        {
            myTextBox.type = TextFieldType.INPUT;
            myTextBox.background = true;
            addChild(myTextBox);
```

```
            myTextBox.text = myText;
            myTextBox.addEventListener(TextEvent.TEXT_INPUT, textInputCapture);
        }

        public function textInputCapture(event:TextEvent):void
        {
            var str:String = myTextBox.text;
            createOutputBox(str);
        }

        public function createOutputBox(str:String):void
        {
            myOutputBox.background = true;
            myOutputBox.x = 200;
            addChild(myOutputBox);
            myOutputBox.text = str;
        }

    }
}
```

Restriction de la saisie de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les champs de texte de saisie sont souvent utilisés dans les formulaires et les boîtes de dialogue des applications. Il peut donc être judicieux de limiter le type de caractères que l'utilisateur peut saisir, ou même de masquer la saisie (pour un mot de passe par exemple). La classe flash.text.TextField possède une propriété `displayAsPassword` et une propriété `restrict` qui permettent de contrôler la saisie par l'utilisateur.

La propriété `displayAsPassword` masque simplement le texte (en l'affichant sous forme d'astérisques) à mesure que l'utilisateur le saisit. Lorsque `displayAsPassword` a la valeur `true`, les commandes Couper et Copier, ainsi que les raccourcis clavier correspondants ne fonctionnent pas. Comme le montre l'exemple suivant, vous pouvez attribuer la propriété `displayAsPassword`, comme vous le feriez pour des propriétés d'arrière-plan et de couleur :

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

La propriété `restrict` est légèrement plus compliquée, puisque vous devez spécifier les caractères que l'utilisateur peut saisir dans le champ de texte. Il est possible d'autoriser la saisie de lettres spécifiques et de nombres, mais aussi de plages de lettres, de nombres et de caractères. Le code ci-après permet à l'utilisateur de saisir uniquement des lettres majuscules (pas de nombres, ni de caractères spéciaux) dans le champ de texte :

```
myTextBox.restrict = "A-Z";
```

ActionScript 3.0 utilise le tiret pour définir les séries et le caractère circonflexe pour exclure des caractères. Pour plus d'informations sur la définition de restrictions associées à un champ de texte de saisie, voir l'entrée `flash.text.TextField.restrict` dans le Guide de référence ActionScript 3.0 pour Flash Professional.

Remarque : *si vous utilisez la propriété* `flash.text.TextField.restrict`*, le moteur d'exécution convertit automatiquement les lettres restreintes en caractères de casse autorisée. Si vous utilisez la propriété* `fl.text.TLFTextField.restrict` *(c'est-à-dire si vous utilisez un champ de texte TLF), le moteur d'exécution ignore les lettres restreintes.*

Mise en forme du texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Plusieurs options permettent de programmer la mise en forme du texte à afficher. Vous pouvez définir ses propriétés directement dans l'occurrence de TextField, par exemple `TextFIeld.thickness`, `TextField.textColor` et `TextField.textHeight`.Vous pouvez aussi désigner le contenu du champ de texte à l'aide de la propriété `htmlText` et utiliser des balises HTML prises en charge, telles que `b`, `i` et `u`. Vous pouvez enfin appliquer des objets TextFormat aux champs de texte contenant du texte brut, ou des objets StyleSheet aux champs contenant la propriété `htmlText`. Les objets TextFormat et StyleSheet offrent un meilleur contrôle et davantage de cohérence sur l'aspect du texte pour l'ensemble de l'application. Il est possible de définir un objet TextFormat ou StyleSheet et de l'appliquer à une partie ou à l'ensemble des champs de texte de l'application.

Attribution de formats texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe TextFormat permet de définir différentes propriétés d'affichage du texte et de les appliquer à tout le contenu d'un objet TextField, ou à une plage de texte.

L'exemple suivant applique un objet TextFormat à un objet TextField complet, puis un second objet TextFormat à une plage de texte de cet objet TextField :

```
var tf:TextField = new TextField();
tf.text = "Hello Hello";

var format1:TextFormat = new TextFormat();
format1.color = 0xFF0000;

var format2:TextFormat = new TextFormat();
format2.font = "Courier";

tf.setTextFormat(format1);
var startRange:uint = 6;
tf.setTextFormat(format2, startRange);

addChild(tf);
```

La méthode `TextField.setTextFormat()` n'affecte que le texte qui est déjà affiché dans le champ de texte. Si le contenu de l'objet TextField change, il peut être nécessaire d'appeler à nouveau la méthode `TextField.setTextFormat()` pour ré-appliquer la mise en forme. Vous pouvez également utiliser la propriété `defaultTextFormat` de TextField pour spécifier le format à utiliser pour le texte saisi par l'utilisateur.

Application de feuilles de style en cascade

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les champs de texte peuvent contenir du texte brut ou du texte au format HTML. Le texte brut est stocké dans la propriété `text` de l'occurrence, et le texte HTML dans la propriété `htmlText`.

Vous pouvez utiliser des déclarations de styles CSS pour définir des styles de texte à appliquer ensuite à différents champs de texte. Une déclaration de style CSS peut être créée par code ou chargée lors de l'exécution à partir d'un fichier CSS externe.

C'est la classe flash.text.StyleSheet qui gère les styles CSS. La classe StyleSheet ne reconnaît qu'un nombre limité de propriétés CSS. La liste détaillée des propriétés de style prises en charge par la classe StyleSheet figure dans l'entrée flash.textStylesheet du Guide de référence ActionScript 3.0 pour Flash Professional.

Comme le montre l'exemple suivant, vous pouvez créer des feuilles de style CSS et les appliquer à du texte HTML au moyen de l'objet StyleSheet :

```
var style:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.fontSize = "bold";
styleObj.color = "#FF0000";
style.setStyle(".darkRed", styleObj);

var tf:TextField = new TextField();
tf.styleSheet = style;
tf.htmlText = "<span class = 'darkRed'>Red</span> apple";

addChild(tf);
```

Après la création de l'objet StyleSheet, le code crée un objet simple pour contenir un jeu de propriétés de déclaration de style. Il appelle ensuite la méthode `StyleSheet.setStyle()`, qui ajoute le nouveau style à la feuille de style sous le nom « .darkred ». Puis il applique les formats de la feuille de styles en affectant l'objet StyleSheet à la propriété `styleSheet` de TextField.

Pour que les styles CSS puissent prendre effet, il est nécessaire d'appliquer la feuille de style à l'objet TextField avant de définir la propriété `htmlText`.

Par essence, un champ de texte doté d'une feuille de style n'est pas modifiable. Si vous attribuez une feuille de style à un champ de texte de saisie, le champ de texte affiche les propriétés de la feuille de style, mais le champ de texte ne permet pas à l'utilisateur de saisir du texte. En outre, vous ne pouvez pas utiliser les méthodes ActionScript suivantes sur un champ de texte doté d'une feuille de style :

- La méthode `TextField.replaceText()`
- La méthode `TextField.replaceSelectedText()`
- La propriété `TextField.defaultTextFormat`
- La méthode `TextField.setTextFormat()`

Si un champ de texte est doté d'une feuille de style mais que par la suite la propriété `TextField.styleSheet` reçoit la valeur `null`, `TextField.text` et `TextField.htmlText` ajoutent des balises et des attributs à leurs contenus afin d'incorporer le formatage de la feuille de style précédemment attribuée. Pour préserver la propriété `htmlText` d'origine, enregistrez-la dans une variable avant d'attribuer la valeur `null` à la feuille de style.

Chargement de fichiers CSS externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'utilisation de feuilles de style CSS pour la mise en forme offre plus de possibilités s'il est possible de charger les informations de CSS à partir d'un fichier externe lors de l'exécution. Si les données CSS sont externes à l'application, il est possible de changer le style visuel du texte sans devoir modifier le code source ActionScript 3.0. En effet, après le déploiement de l'application, vous pouvez encore modifier le fichier CSS externe pour obtenir un nouvel aspect, sans devoir redéployer le fichier SWF de l'application.

La méthode `StyleSheet.parseCSS()` convertit une chaîne contenant des données CSS en déclarations de style dans l'objet StyleSheet. L'exemple suivant montre comment lire un fichier CSS externe et appliquer ses déclarations de style à un objet TextField.

Voici le contenu du fichier CSS à charger. Il est appelé « example.css » :

```
p {
    font-family: Times New Roman, Times, _serif;
    font-size: 14;
}

h1 {
    font-family: Arial, Helvetica, _sans;
    font-size: 20;
    font-weight: bold;
}

.bluetext {
    color: #0000CC;
}
```

Voici maintenant le code ActionScript d'une classe qui charge le fichier example.css et en applique les styles au contenu de l'objet TextField :

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
    import flash.text.StyleSheet;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;

    public class CSSFormattingExample extends Sprite
    {
        var loader:URLLoader;
        var field:TextField;
        var exampleText:String = "<h1>This is a headline</h1>" +
            "<p>This is a line of text. <span class='bluetext'>" +
            "This line of text is colored blue.</span></p>";

        public function CSSFormattingExample():void
        {
            field = new TextField();
            field.width = 300;
```

```
            field.autoSize = TextFieldAutoSize.LEFT;
            field.wordWrap = true;
            addChild(field);

            var req:URLRequest = new URLRequest("example.css");

            loader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, onCSSFileLoaded);
            loader.load(req);
        }

        public function onCSSFileLoaded(event:Event):void
        {
            var sheet:StyleSheet = new StyleSheet();
            sheet.parseCSS(loader.data);
            field.styleSheet = sheet;
            field.htmlText = exampleText;
        }
    }
}
```

Lorsque les données de CSS sont chargées, la méthode `onCSSFileLoaded()` s'exécute et appelle la méthode `StyleSheet.parseCSS()` pour transférer les déclarations de style à l'objet StyleSheet.

Formatage de plages de texte au sein d'un champ de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe flash.text.TextField contient une méthode particulièrement utile, `setTextFormat()`. La méthode `setTextFormat()` permet d'affecter des propriétés spécifiques à une partie du contenu d'un champ de texte en réponse à une action de l'utilisateur, par exemple pour rappeler à l'utilisateur que certains champs d'un formulaire doivent être renseignés, ou encore pour changer l'aspect d'un passage de texte si l'utilisateur sélectionne une partie de ce texte.

L'exemple suivant utilise la méthode `TextField.setTextFormat()` sur une plage de caractères pour modifier l'aspect d'une partie du contenu de `myTextField` lorsque l'utilisateur clique dans ce champ de texte :

```
var myTextField:TextField = new TextField();
myTextField.text = "No matter where you click on this text field the TEXT IN ALL CAPS changes
format.";
myTextField.autoSize = TextFieldAutoSize.LEFT;
addChild(myTextField);
addEventListener(MouseEvent.CLICK, changeText);

var myformat:TextFormat = new TextFormat();
myformat.color = 0xFF0000;
myformat.size = 18;
myformat.underline = true;

function changeText(event:MouseEvent):void
{
    myTextField.setTextFormat(myformat, 49, 65);
}
```

Fonctions avancées d'affichage de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le package flash.text d'ActionScript 3.0 offre plusieurs classes qui permettent de contrôler les propriétés du texte affiché, notamment les polices intégrées, les paramètres d'anticrènelage, le canal alpha et autres paramètres spécifiques. Le Guide de référence ActionScript 3.0 pour Flash Professional fournit des descriptions détaillées de ces classes et de leurs propriétés, notamment des classes CSMSettings, Font et TextRenderer.

Utilisation de polices incorporées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous spécifiez une police précise pour un objet TextField de votre application, Flash Player ou AIR recherche une police résidente du même nom sur l'ordinateur de l'utilisateur. Si cette police n'est pas chargée sur cet ordinateur, ou s'il existe une police de ce nom mais dans une version légèrement différente, le texte peut apparaître très différent de ce que vous aviez prévu. Par défaut, le texte s'affiche dans la police Times Roman.

Pour que l'utilisateur voie exactement la police voulue, vous pouvez incorporer cette police dans le fichier SWF de votre application. Les polices intégrées présentent de nombreux avantages :

- Les caractères des polices incorporées sont anticrènelés, ce qui les rend plus agréables à lire, en particulier pour les grandes tailles de texte.
- Il est possible de faire pivoter les polices incorporées.
- Il est possible de rendre transparent ou semi-transparent le texte des polices incorporées.
- Il est possible d'utiliser le style CSS `kerning` (crénage) avec les polices incorporées.

Le principal inconvénient des polices incorporées est l'augmentation de la taille du fichier de l'application.

La méthode exacte à utiliser pour intégrer un fichier de police dans le fichier SWF de l'application varie selon l'environnement de développement.

Une fois la police intégrée, il est possible de faire en sorte que l'objet TextField utilise la police correcte :

- Mettez la propriété `embedFonts` de l'objet TextField sur `true`.
- Créez un objet TextFormat, donnez à sa propriété `fontFamily` le nom de la police incorporée, et appliquez l'objet TextFormat au TextField. Dans le cas d'une police incorporée, la propriété `fontFamily` ne doit contenir qu'un seul nom. Elle ne peut pas utiliser une liste de polices séparées par des virgules.
- Si vous utilisez des styles CSS pour les polices d'objets TextFields, donnez à la propriété CSS `font-family` le nom de la police incorporée. Si vous voulez utiliser une police incorporée, la propriété `font-family` ne doit contenir qu'un seul nom, et non pas une liste de noms.

Intégration d'une police dans Flash

Flash Professional vous permet d'intégrer pratiquement toutes les polices installées sur votre système, notamment les polices TrueType et les polices Postscript Type 1.

Il existe plusieurs façons d'intégrer des polices dans une application. Vous pouvez par exemple :

- définir la police et les propriétés de style d'un objet TextField sur la Scène, puis en cocher la case Incorporer les polices ;
- créer et référencer un symbole de police ;

- créer et utiliser une bibliothèque d'exécution partagée contenant les symboles de la police intégrée.

Pour plus d'informations sur l'intégration de polices dans les applications, voir « Polices intégrées pour champs de texte dynamique ou de saisie » dans *Utilisation de Flash*.

Intégration d'une police dans Flex

Il existe plusieurs façons d'intégrer des polices dans une application Flex. Vous pouvez par exemple :

- utiliser la balise de métadonnées `[Embed]` dans un script ;
- utiliser la déclaration de style `@font-face` ;
- définir la classe de la police et l'intégrer par le biais de la balise `[Embed]`.

Seules les polices TrueType peuvent directement être intégrées dans une application Flex. Les polices dans un autre format, telles que les polices Postscript Type 1, doivent tout d'abord être intégrées dans un fichier SWF à l'aide de Flash Professional ; vous pouvez ensuite utiliser ce fichier SWF dans votre application Flex. Pour plus d'informations sur l'utilisation de polices intégrées dans Flex à partir de fichiers SWF, voir « Intégration de polices à partir de fichiers SWF » dans le manuel *Utilisation de Flex 4*.

Voir aussi

Incorporation de polices pour assurer la cohérence de l'apparence du texte

Peter deHaan : Embedding fonts (disponible en anglais uniquement)

Divillysausages.com : AS3 Font embedding masterclass (disponible en anglais uniquement)

Contrôle de la netteté, de l'épaisseur et de l'anticrènelage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, Flash Player ou AIR détermine les paramètres de contrôle d'affichage du texte (netteté, épaisseur et anticrènelage) qui s'appliquent lorsque le texte change de taille et de couleur ou s'affiche sur différents arrière-plans. Dans certains cas, vous pouvez définir ces paramètres, par exemple si le texte est très petit ou très gros, ou s'il s'affiche sur plusieurs arrière-plans. La classe `flash.text.TextRenderer` et les classes associées, telles que CSMSettings, permettent de remplacer les paramètres de Flash Player ou d'AIR. Elles offrent un contrôle précis de la qualité d'affichage du texte incorporé. Pour plus d'informations sur les polices intégrées, voir « Utilisation de polices incorporées » à la page 397.

Remarque : *la propriété .antiAliasType de la classe* `flash.text.TextField` *doit avoir la valeur* `AntiAliasType.ADVANCED` *pour que vous puissiez définir la netteté, l'épaisseur ou la propriété gridFitType, ou pour que vous puissiez utiliser la méthode* `TextRenderer.setAdvancedAntiAliasingTable()`.

L'exemple suivant applique des propriétés personnalisées de modulation continue du trait (CSM) et de mise en forme au texte affiché, en utilisant la police incorporée `myFont`. Lorsque l'utilisateur clique sur le texte affiché, Flash Player ou Adobe AIR applique ces paramètres personnalisés :

```
var format:TextFormat = new TextFormat();
format.color = 0x336699;
format.size = 48;
format.font = "myFont";

var myText:TextField = new TextField();
myText.embedFonts = true;
myText.autoSize = TextFieldAutoSize.LEFT;
myText.antiAliasType = AntiAliasType.ADVANCED;
myText.defaultTextFormat = format;
myText.selectable = false;
myText.mouseEnabled = true;
myText.text = "Hello World";
addChild(myText);
myText.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:Event):void
{
    var myAntiAliasSettings = new CSMSettings(48, 0.8, -0.8);
    var myAliasTable:Array = new Array(myAntiAliasSettings);
    TextRenderer.setAdvancedAntiAliasingTable("myFont", FontStyle.ITALIC,
TextColorType.DARK_COLOR, myAliasTable);
}
```

Utilisation du texte statique

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le texte statique est créé dans Flash Professional uniquement. Il est impossible d'instancier du texte statique en ActionScript. Le texte statique est utile si le contenu textuel est court et ne doit pas changer (contrairement au texte dynamique). Vous pouvez assimiler le texte statique à un élément graphique comparable à un cercle ou un carré dessiné sur la Scène dans Flash Professional. Bien que le texte statique offre moins de possibilités que le texte dynamique, ActionScript 3.0 permet de lire les valeurs des propriétés du texte statique à l'aide de la classe StaticText. En outre, vous pouvez utiliser la classe TextSnapshot pour extraire des valeurs du texte statique.

Accès aux champs de texte statique à l'aide de la classe StaticText

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous utilisez généralement la classe flash.text.StaticText dans le panneau Actions de Flash Professional pour agir sur une occurrence de texte statique placée sur la scène. Vous pouvez également travailler dans des fichiers ActionScript qui interagissent avec le fichier SWF contenant le texte statique. Dans les deux cas, il est impossible de créer par code une occurrence de texte statique. Le texte statique est créé dans Flash Professional.

Pour créer une référence à un champ de texte statique existant, vous pouvez effectuer une itération sur les éléments de la liste d'affichage et affecter une variable. Exemple :

```
for (var i = 0; i < this.numChildren; i++) {
var displayitem:DisplayObject = this.getChildAt(i);
if (displayitem instanceof StaticText) {
trace("a static text field is item " + i + " on the display list");
        var myFieldLabel:StaticText = StaticText(displayitem);
        trace("and contains the text: " + myFieldLabel.text);
}
}
```

Lorsque vous disposez d'une référence à un champ de texte statique, vous pouvez utiliser les propriétés de ce champ dans ActionScript 3.0. Le code suivant est associé à une image du scénario et suppose qu'une variable appelée `myFieldLabel` est affectée à une référence à un champ de texte statique. Un champ de texte dynamique `myField` est positionné relativement aux valeurs `x` et `y` de `myFieldLabel` et affiche à nouveau la valeur de `myFieldLabel`.

```
var myField:TextField = new TextField();
addChild(myField);
myField.x = myFieldLabel.x;
myField.y = myFieldLabel.y + 20;
myField.autoSize = TextFieldAutoSize.LEFT;
myField.text = "and " + myFieldLabel.text
```

Utilisation de la classe TextSnapshot

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour travailler par programmation avec une occurrence de texte statique existante, utilisez la classe flash.text.TextSnapshot pour modifier la propriété `textSnapshot` d'un objet flash.display.DisplayObjectContainer. En d'autres termes, créez une occurrence de TextSnapshot à partir de la propriété `DisplayObjectContainer.textSnapshot`. Vous pouvez alors appliquer des méthodes à cette occurrence afin d'extraire des valeurs ou de sélectionner des portions du texte statique.

Par exemple, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la scène. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var mySnap:TextSnapshot = this.textSnapshot;
var count:Number = mySnap.charCount;
mySnap.setSelected(0, 4, true);
mySnap.setSelected(1, 2, false);
var myText:String = mySnap.getSelectedText(false);
trace(myText);
```

La classe TextSnapshot est utile pour extraire le texte des champs de texte statique dans un fichier SWF chargé, au cas où vous souhaiteriez utiliser ce texte comme valeur dans une autre zone de l'application.

Exemple TextField : mise en forme du texte dans le style « article de journal »

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple « Article de journal » met en forme le texte pour lui donner l'aspect d'un article de journal. Le texte saisi peut contenir un gros titre, un intertitre et le corps de l'article. En fonction d'une largeur et d'une hauteur d'affichage, cet exemple met en forme le gros titre et l'intertitre pour qu'ils occupent toute la largeur disponible. Le texte de l'article est réparti sur plusieurs colonnes.

Cet exemple illustre les techniques de programmation en ActionScript suivantes :

- Extension de la classe TextField
- Chargement et application d'un fichier CSS externe
- Conversion de styles CSS en objets TextFormat
- Utilisation de la classe TextLineMetrics pour obtenir des informations sur la taille d'affichage du texte

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application News Layout se trouvent dans le dossier Samples/NewsLayout. L'application se compose des fichiers suivants :

Fichier	Description
NewsLayout.mxml ou NewsLayout.fla	Interface utilisateur de l'application pour Flex (MXML) ou Flash (FLA).
com/example/programmingas3/newslayout/StoryLayoutComponent.as	Classe Flex UIComponent qui place l'occurrence de StoryLayout.
com/example/programmingas3/newslayout/StoryLayout.as	Principale classe ActionScript chargée d'organiser les composants d'un article pour leur affichage.
com/example/programmingas3/newslayout/FormattedTextField.as	Sous-classe de la classe TextField qui gère son propre objet TextFormat.
com/example/programmingas3/newslayout/HeadlineTextField.as	Sous-classe de la classe FormattedTextField qui ajuste la taille des polices en fonction de la largeur voulue.
com/example/programmingas3/newslayout/MultiColumnTextField.as	Classe ActionScript qui répartit le texte de l'article sur plusieurs colonnes.
story.css	Fichier CSS définissant les styles du texte pour la mise en page.

Lecture du fichier CSS externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application News Layout commence par récupérer le texte de l'article à partir d'un fichier XML local. Elle lit ensuite un fichier CSS externe contenant les informations de mise en forme pour le gros titre, l'intertitre et le texte.

Ce fichier CSS définit trois styles, un style de paragraphe standard pour l'article et les styles h1 et h2, respectivement pour le gros titre et l'intertitre.

```
p {
    font-family: Georgia, "Times New Roman", Times, _serif;
    font-size: 12;
    leading: 2;
    text-align: justify;
    indent: 24;
}

h1 {
    font-family: Verdana, Arial, Helvetica, _sans;
    font-size: 20;
    font-weight: bold;
    color: #000099;
    text-align: left;
}

h2 {
    font-family: Verdana, Arial, Helvetica, _sans;
    font-size: 16;
    font-weight: normal;
    text-align: left;
}
```

La technique utilisée pour lire le fichier CSS externe est identique à celle décrite à la section « Chargement de fichiers CSS externes » à la page 395. Après le chargement du fichier CSS, l'application exécute la méthode `onCSSFileLoaded()`, représentée ci-dessous.

```
public function onCSSFileLoaded(event:Event):void
{
    this.sheet = new StyleSheet();
    this.sheet.parseCSS(loader.data);

    h1Format = getTextStyle("h1", this.sheet);
    if (h1Format == null)
    {
        h1Format = getDefaultHeadFormat();
    }
    h2Format = getTextStyle("h2", this.sheet);
    if (h2Format == null)
    {
        h2Format = getDefaultHeadFormat();
        h2Format.size = 16;
    }
    pFormat = getTextStyle("p", this.sheet);
    if (pFormat == null)
    {
        pFormat = getDefaultTextFormat();
        pFormat.size = 12;
    }
    displayText();
}
```

La méthode `onCSSFileLoaded()` crée un objet StyleSheet qui analyse les données du fichier CSS. Le texte principal de l'article est affiché dans un objet MultiColumnTextField, qui peut utiliser directement un objet StyleSheet. Toutefois, les champs du gros titre utilisent la classe HeadlineTextField, qui utilise un objet TextFormat pour sa mise en forme.

La méthode `onCSSFileLoaded()` appelle deux fois la méthode `getTextStyle()` pour convertir une déclaration de style CSS en un objet TextFormat destiné aux deux objets HeadlineTextField.

```
public function getTextStyle(styleName:String, ss:StyleSheet):TextFormat
{
    var format:TextFormat = null;

    var style:Object = ss.getStyle(styleName);
    if (style != null)
    {
        var colorStr:String = style.color;
        if (colorStr != null && colorStr.indexOf("#") == 0)
        {
            style.color = colorStr.substr(1);
        }
        format = new TextFormat(style.fontFamily,
                        style.fontSize,
                        style.color,
                        (style.fontWeight == "bold"),
                        (style.fontStyle == "italic"),
                        (style.textDecoration == "underline"),
                        style.url,
                        style.target,
                        style.textAlign,
                        style.marginLeft,
                        style.marginRight,
                        style.indent,
                        style.leading);

        if (style.hasOwnProperty("letterSpacing"))
        {
            format.letterSpacing = style.letterSpacing;
        }
    }
    return format;
}
```

Les noms de propriétés et la signification de leurs valeurs diffèrent entre les déclarations de style CSS et les objets TextFormat. La méthode `getTextStyle()` transforme donc les valeurs des propriétés CSS dans les valeurs attendues par l'objet TextFormat.

Disposition des éléments de l'article sur la page

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe StoryLayout formate et met en page les champs de texte dévolus au gros titre, à l'intertitre et à l'article dans le style d'une page de journal. La méthode `displayText()` crée et place les divers champs.

```
public function displayText():void
{
    headlineTxt = new HeadlineTextField(h1Format);
    headlineTxt.wordWrap = true;
    headlineTxt.x = this.paddingLeft;
    headlineTxt.y = this.paddingTop;
    headlineTxt.width = this.preferredWidth;
    this.addChild(headlineTxt);

    headlineTxt.fitText(this.headline, 1, true);

    subtitleTxt = new HeadlineTextField(h2Format);
    subtitleTxt.wordWrap = true;
    subtitleTxt.x = this.paddingLeft;
    subtitleTxt.y = headlineTxt.y + headlineTxt.height;
    subtitleTxt.width = this.preferredWidth;
    this.addChild(subtitleTxt);

    subtitleTxt.fitText(this.subtitle, 2, false);

    storyTxt = new MultiColumnText(this.numColumns, 20,
                        this.preferredWidth, 400, true, this.pFormat);
    storyTxt.x = this.paddingLeft;
    storyTxt.y = subtitleTxt.y + subtitleTxt.height + 10;
    this.addChild(storyTxt);

    storyTxt.text = this.content;
...
```

Chaque champ est placé sous le champ précédent en effectuant le calcul suivant : la propriété `y` du champ est égale à la propriété `y` du champ précédent, plus sa hauteur. Ce calcul dynamique de position est nécessaire, car les objets HeadlineTextField et MultiColumnTextField peuvent changer de hauteur en fonction de leur contenu.

Modification de la taille de la police en fonction de la taille du champ

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Sur la base d'une largeur en pixels et d'un nombre maximum de lignes à afficher, l'objet HeadlineTextField modifie la taille de la police pour adapter le texte aux dimensions du champ. Si le texte est court, la police est de grande taille, créant ainsi un gros titre de style tabloïde. Si le texte est long, la police est de plus petite taille.

La méthode `HeadlineTextField.fitText()` reproduite ci-dessous est chargée du redimensionnement du texte :

```
public function fitText(msg:String, maxLines:uint = 1, toUpper:Boolean = false,
targetWidth:Number = -1):uint
{
    this.text = toUpper ? msg.toUpperCase() : msg;

    if (targetWidth == -1)
    {
        targetWidth = this.width;
    }

    var pixelsPerChar:Number = targetWidth / msg.length;

    var pointSize:Number = Math.min(MAX_POINT_SIZE, Math.round(pixelsPerChar * 1.8 * maxLines));

    if (pointSize < 6)
    {
        // the point size is too small
        return pointSize;
    }

    this.changeSize(pointSize);

    if (this.numLines > maxLines)
    {
        return shrinkText(--pointSize, maxLines);
    }
    else
    {
        return growText(pointSize, maxLines);
    }
}

public function growText(pointSize:Number, maxLines:uint = 1):Number
{
    if (pointSize >= MAX_POINT_SIZE)
    {
        return pointSize;
    }

    this.changeSize(pointSize + 1);

    if (this.numLines > maxLines)
    {
        // set it back to the last size
        this.changeSize(pointSize);
        return pointSize;
    }
    else
    {
        return growText(pointSize + 1, maxLines);
    }
```

```
}

public function shrinkText(pointSize:Number, maxLines:uint=1):Number
{
    if (pointSize <= MIN_POINT_SIZE)
    {
        return pointSize;
    }

    this.changeSize(pointSize);

    if (this.numLines > maxLines)
    {
        return shrinkText(pointSize - 1, maxLines);
    }
    else
    {
        return pointSize;
    }
}
```

La méthode `HeadlineTextField.fitText()` utilise une technique récursive simple pour dimensionner le texte. Elle estime d'abord un nombre moyen de pixels par caractère pour le texte, puis calcule une taille de départ. Elle change alors la taille de la police et vérifie si le texte a renvoyé des mots à la ligne et si le nombre maximal de lignes est dépassé. Si c'est le cas, elle appelle la méthode `shrinkText()` pour réduire la taille du texte, et teste à nouveau. Si le nombre de lignes n'est pas trop important, elle appelle la méthode `growText()` pour augmenter la taille du texte, et teste à nouveau. Ce processus s'interrompt lorsque l'augmentation de taille du texte d'un seul point crée un nombre de lignes trop élevé.

Répartition du texte sur plusieurs colonnes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe MultiColumnTextField répartit le texte entre plusieurs objets TextField qui sont organisés comme des colonnes de texte sur une page de journal.

Le constructeur de `MultiColumnTextField()` crée tout d'abord un tableau d'objets TextField, un pour chaque colonne :

```
for (var i:int = 0; i < cols; i++)
{
    var field:TextField = new TextField();
    field.multiline = true;
    field.autoSize = TextFieldAutoSize.NONE;
    field.wordWrap = true;
    field.width = this.colWidth;
    field.setTextFormat(this.format);
    this.fieldArray.push(field);
    this.addChild(field);
}
```

Chaque objet TextField est ajouté au tableau et à la liste d'affichage à l'aide de la méthode `addChild()`.

Si la propriété `text` ou `styleSheet` de StoryLayout change, la méthode `layoutColumns()` est appelée pour réafficher le texte. La méthode `layoutColumns()` appelle la méthode `getOptimalHeight()` pour déterminer la hauteur correcte en pixels nécessaire pour adapter tout le texte en fonction de la largeur disponible.

```
public function getOptimalHeight(str:String):int
{
    if (field.text == "" || field.text == null)
    {
        return this.preferredHeight;
    }
    else
    {
        this.linesPerCol = Math.ceil(field.numLines / this.numColumns);

        var metrics:TextLineMetrics = field.getLineMetrics(0);
        this.lineHeight = metrics.height;
        var prefHeight:int = linesPerCol * this.lineHeight;

        return prefHeight + 4;
    }
}
```

La méthode `getOptimalHeight()` calcule d'abord la largeur de chaque colonne. Elle définit ensuite la propriété `htmlText` du premier objet TextField du tableau. La méthode `getOptimalHeight()` utilise ce premier objet TextField pour connaître le nombre total de lignes renvoyées à la ligne, et en déduit donc le nombre de lignes optimal pour chaque colonne. Elle appelle ensuite la méthode `TextField.getLineMetrics()` pour obtenir un objet TextLineMetrics contenant des informations sur la taille du texte de la première ligne. La propriété `TextLineMetrics.height` représente la hauteur totale (en pixels) d'une ligne de texte, avec les mesures ascendantes, descendantes et l'interligne. La hauteur optimale de l'objet MultiColumnTextField est donc la hauteur de ligne multipliée par le nombre de lignes par colonne, plus 4 pour tenir compte de la bordure de deux pixels en haut et en bas de l'objet TextField.

Voici le code complet de la méthode `layoutColumns()` :

```
public function layoutColumns():void
{
    if (this._text == "" || this._text == null)
    {
        return;
    }

    var field:TextField = fieldArray[0] as TextField;
    field.text = this._text;
    field.setTextFormat(this.format);

    this.preferredHeight = this.getOptimalHeight(field);

    var remainder:String = this._text;
    var fieldText:String = "";
    var lastLineEndedPara:Boolean = true;

    var indent:Number = this.format.indent as Number;

    for (var i:int = 0; i < fieldArray.length; i++)
    {
        field = this.fieldArray[i] as TextField;

        field.height = this.preferredHeight;
        field.text = remainder;

        field.setTextFormat(this.format);
```

```
var lineLen:int;
if (indent > 0 && !lastLineEndedPara && field.numLines > 0)
{
    lineLen = field.getLineLength(0);
    if (lineLen > 0)
    {
        field.setTextFormat(this.firstLineFormat, 0, lineLen);
        }
    }

field.x = i * (colWidth + gutter);
field.y = 0;

remainder = "";
fieldText = "";

var linesRemaining:int = field.numLines;
var linesVisible:int = Math.min(this.linesPerCol, linesRemaining);

for (var j:int = 0; j < linesRemaining; j++)
{
    if (j < linesVisible)
    {
        fieldText += field.getLineText(j);
    }
    else
    {
        remainder +=field.getLineText(j);
    }
}

field.text = fieldText;

field.setTextFormat(this.format);

if (indent > 0 && !lastLineEndedPara)
{
    lineLen = field.getLineLength(0);
    if (lineLen > 0)
    {
        field.setTextFormat(this.firstLineFormat, 0, lineLen);
    }
}

var lastLine:String = field.getLineText(field.numLines - 1);
var lastCharCode:Number = lastLine.charCodeAt(lastLine.length - 1);
```

```
        if (lastCharCode == 10 || lastCharCode == 13)
        {
        lastLineEndedPara = true;
        }
        else
        {
        lastLineEndedPara = false;
        }

        if ((this.format.align == TextFormatAlign.JUSTIFY) &&
                (i < fieldArray.length - 1))
        {
        if (!lastLineEndedPara)
        {
            justifyLastLine(field, lastLine);
        }
    }
    }
}
```

Lorsque la propriété `preferredHeight` a été définie par un appel à la méthode `getOptimalHeight`, la méthode `layoutColumns()` parcourt les objets TextField et définit la hauteur de chacun avec la valeur `preferredHeight`. La méthode `layoutColumns()` distribue ensuite le nombre de lignes de texte adapté à chaque champ, de sorte qu'aucun défilement ne se produise dans l'un d'eux et que le texte de chaque champ enchaîne sur celui du champ précédent. Si le style d'alignement du texte a été défini sur « justifier », la méthode `justifyLastLine()` est appelée pour justifier la ligne finale du texte dans un champ. Dans le cas contraire, la dernière ligne est considérée comme une ligne de fin de paragraphe et n'est donc pas justifiée.

Chapitre 22 : Utilisation de Flash Text Engine

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Adobe® Flash® Text Engine (FTE), disponible à partir de Flash Player 10 et Adobe® AIR™1.5, propose une prise en charge de bas niveau pour un contrôle sophistiqué des mesures de texte, de la mise en forme et du texte bidirectionnel. Il se caractérise par un flux de texte optimisé et une prise en charge des langues enrichie. Bien qu'il puisse être utilisé pour créer et gérer de simples éléments de texte, FTE est l'outil de base pour les développeurs qui souhaitent créer des composants d'édition de texte. En tant que tel, Flash Text Engine nécessite des connaissances avancées en programmation. Pour afficher des éléments de texte simples, voir « Utilisation de la classe TextField » à la page 385.

Text Layout Framework (TLF), qui comprend un composant de manipulation de texte basé sur FTE, propose une méthode d'utilisation de ses fonctions avancées plus conviviale. TLF est une bibliothèque extensible reposant entièrement sur ActionScript 3.0. Vous pouvez utiliser le composant TLF existant ou utiliser la structure pour créer votre propre composant de texte. Pour plus d'informations, voir « Utilisation de Text Layout Framework » à la page 440.

Voir aussi

Package flash.text.engine

Création et affichage de texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les classes qui constituent Flash Text Engine vous permettent de créer, de mettre en forme et de contrôler le texte. Les classes suivantes constituent les éléments de base pour la création et l'affichage de texte avec Flash Text Engine :

- TextElement/GraphicElement/GroupElement : renferment le contenu d'une occurrence de TextBlock
- ElementFormat : spécifie les attributs de mise en forme du contenu d'une occurrence de TextBlock
- TextBlock : classe usine pour la création d'un paragraphe de texte
- TextLine : ligne de texte créée à partir de TextBlock

Pour afficher du texte, créez un objet TextElement à partir d'un élément String en utilisant un objet ElementFormat pour définir les caractéristiques de mise en forme. Affectez l'objet TextElement à la propriété `content` d'un objet TextBlock. Créez les lignes de texte à afficher en appelant la méthode `TextBlock.createTextLine()`. La méthode `createTextLine()` renvoie un objet TextLine qui contient le nombre de caractères de la chaîne correspondant à la largeur spécifiée. Appelez plusieurs fois la méthode jusqu'à ce que la chaîne entière s'affiche sous forme de lignes. Une fois toutes les lignes créées, la valeur `TextLineCreationResult.COMPLETE` est affectée à la propriété textLineCreationResult de l'objet TextBlock. Pour afficher les lignes, ajoutez-les à la liste d'affichage (associées aux valeurs de position `x` et `y` appropriées).

Par exemple, le code suivant utilise ces classes FTE pour afficher « Hello World! This is Flash Text Engine! ».à l'aide du format et de la police par défaut. Dans cet exemple simple, une seule ligne de texte est générée.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class HelloWorldExample extends Sprite
    {
        public function HelloWorldExample()
        {
            var str = "Hello World! This is Flash Text Engine!";
            var format:ElementFormat = new ElementFormat();
            var textElement:TextElement = new TextElement(str, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = textElement;

            var textLine1:TextLine = textBlock.createTextLine(null, 300);
            addChild(textLine1);
            textLine1.x = 30;
            textLine1.y = 30;
        }
    }
}
```

Les paramètres requis pour `createTextLine()` spécifient la ligne où doit commencer la nouvelle ligne et la largeur de celle-ci en pixels. La ligne où doit commencer la nouvelle ligne correspond généralement à la ligne précédente, mais dans le cas de la première ligne, la valeur est `null`.

Ajout d'objets GraphicElement et GroupElement

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez affecter un objet GraphicElement à un objet TextBlock en vue d'afficher une image ou un élément graphique. Il vous suffit pour cela de créer une occurrence de la classe GraphicElement à partir d'un graphique ou d'une image, puis d'affecter l'occurrence à la propriété `TextBlock.content`. Créez la ligne de texte en appelant la méthode `TextBlock.createTextline()` selon la procédure habituelle. L'exemple suivant crée deux lignes de texte : une avec un objet GraphicElement, l'autre avec un objet TextElement.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;

    public class GraphicElementExample extends Sprite
    {
        public function GraphicElementExample()
        {
            var str:String = "Beware of Dog!";

            var triangle:Shape = new Shape();
            triangle.graphics.beginFill(0xFF0000, 1);
            triangle.graphics.lineStyle(3);
            triangle.graphics.moveTo(30, 0);
            triangle.graphics.lineTo(60, 50);
            triangle.graphics.lineTo(0, 50);
            triangle.graphics.lineTo(30, 0);
            triangle.graphics.endFill();

            var format:ElementFormat = new ElementFormat();
            format.fontSize = 20;

            var graphicElement:GraphicElement = new GraphicElement(triangle, triangle.width,
triangle.height, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = graphicElement;
            var textLine1:TextLine = textBlock.createTextLine(null, triangle.width);
            textLine1.x = 50;
            textLine1.y = 110;
            addChild(textLine1);

            var textElement:TextElement = new TextElement(str, format);
            textBlock.content = textElement;
            var textLine2 = textBlock.createTextLine(null, 300);
            addChild(textLine2);
            textLine2.x = textLine1.x - 30;
            textLine2.y = textLine1.y + 15;
        }
    }
}
```

Vous pouvez créer un objet GroupElement pour créer un groupe d'objets TextElement et GraphicElement ou d'autres objets GroupElement. Vous pouvez affecter un objet GroupElement à la propriété `content` d'un objet TextBlock. Le paramètre au constructeur `GroupElement()` est un vecteur qui pointe vers le texte, le graphique et les éléments qui constituent le groupe. L'exemple suivant regroupe deux éléments graphiques et un élément de texte, et les affecte comme une unité à un bloc de texte.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;
    import flash.display.Shape;
    import flash.display.Graphics;

    public class GroupElementExample extends Sprite
    {
        public function GroupElementExample()
        {
            var str:String = "Beware of Alligators!";

            var triangle1:Shape = new Shape();
            triangle1.graphics.beginFill(0xFF0000, 1);
            triangle1.graphics.lineStyle(3);
            triangle1.graphics.moveTo(30, 0);
            triangle1.graphics.lineTo(60, 50);
            triangle1.graphics.lineTo(0, 50);
            triangle1.graphics.lineTo(30, 0);
            triangle1.graphics.endFill();

            var triangle2:Shape = new Shape();
            triangle2.graphics.beginFill(0xFF0000, 1);
            triangle2.graphics.lineStyle(3);
            triangle2.graphics.moveTo(30, 0);
            triangle2.graphics.lineTo(60, 50);
            triangle2.graphics.lineTo(0, 50);
            triangle2.graphics.lineTo(30, 0);
            triangle2.graphics.endFill();

            var format:ElementFormat = new ElementFormat();
            format.fontSize = 20;
            var graphicElement1:GraphicElement = new GraphicElement(triangle1,
triangle1.width, triangle1.height, format);
            var textElement:TextElement = new TextElement(str, format);
            var graphicElement2:GraphicElement = new GraphicElement(triangle2,
triangle2.width, triangle2.height, format);
            var groupVector:Vector.<ContentElement> = new Vector.<ContentElement>();
            groupVector.push(graphicElement1, textElement, graphicElement2);
            var groupElement = new GroupElement(groupVector);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = groupElement;
            var textLine:TextLine = textBlock.createTextLine(null, 800);
            addChild(textLine);
            textLine.x = 100;
            textLine.y = 200;
        }
    }
}
```

Remplacement du texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez remplacer du texte dans une occurrence de TextBlock en appelant la méthode `TextElement.replaceText()` en vue de remplacer le texte dans l'objet TextElement que vous avez affecté à la propriété `TextBlock.content`.

L'exemple suivant utilise `replaceText()` pour d'abord insérer du texte au début de la ligne, puis pour ajouter du texte à la fin de la ligne et, enfin, pour remplacer du texte au milieu de la ligne.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class ReplaceTextExample extends Sprite
    {
        public function ReplaceTextExample()
        {

            var str:String = "Lorem ipsum dolor sit amet";
            var fontDescription:FontDescription = new FontDescription("Arial");
            var format:ElementFormat = new ElementFormat(fontDescription);
            format.fontSize = 14;
            var textElement:TextElement = new TextElement(str, format);
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = textElement;
            createLine(textBlock, 10);
            textElement.replaceText(0, 0, "A text fragment: ");
            createLine(textBlock, 30);
            textElement.replaceText(43, 43, "...");
            createLine(textBlock, 50);
            textElement.replaceText(23, 28, "(ipsum)");
            createLine(textBlock, 70);
        }

        function createLine(textBlock:TextBlock, y:Number):void {
            var textLine:TextLine = textBlock.createTextLine(null, 300);
            textLine.x = 10;
            textLine.y = y;
            addChild(textLine);
        }
    }
}
```

La méthode `replaceText()` remplace le texte spécifié à l'aide des paramètres `beginIndex` et `endIndex` par le texte spécifié à l'aide du paramètre `newText`. Si les valeurs des paramètres `beginIndex` et `endIndex` sont les mêmes, la méthode `replaceText()` insère le texte spécifié à cet emplacement. Dans le cas contraire, elle remplace les caractères spécifiés à l'aide des paramètres `beginIndex` et `endIndex` par le nouveau texte.

Gestion des événements dans FTE

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Comme c'est le cas pour d'autres objets d'affichage, vous pouvez ajouter des écouteurs d'événement à une occurrence de TextLine. Par exemple, vous pouvez savoir à quel moment un utilisateur passe la souris sur une ligne de texte ou clique sur la ligne. L'exemple suivant détecte ces deux événements. Lorsque vous passez la souris sur une ligne, le curseur prend la forme d'un bouton et lorsque vous cliquez sur la ligne, il change de couleur.

```
package
{
    import flash.text.engine.*;
    import flash.ui.Mouse;
    import flash.display.Sprite
    import flash.events.MouseEvent;
    import flash.events.EventDispatcher;

    public class EventHandlerExample extends Sprite
    {
        var textBlock:TextBlock = new TextBlock();

        public function EventHandlerExample():void
        {
            var str:String = "I'll change color if you click me.";
            var fontDescription:FontDescription = new FontDescription("Arial");
            var format:ElementFormat = new ElementFormat(fontDescription, 18);
            var textElement = new TextElement(str, format);
            textBlock.content = textElement;
            createLine(textBlock);
        }

        private function createLine(textBlock:TextBlock):void
        {
            var textLine:TextLine = textBlock.createTextLine(null, 500);
            textLine.x = 30;
            textLine.y = 30;
            addChild(textLine);
            textLine.addEventListener("mouseOut", mouseOutHandler);
            textLine.addEventListener("mouseOver", mouseOverHandler);
            textLine.addEventListener("click", clickHandler);
        }

        private function mouseOverHandler(event:MouseEvent):void
        {
            Mouse.cursor = "button";
        }

        private function mouseOutHandler(event:MouseEvent):void
        {
            Mouse.cursor = "arrow";
        }

        function clickHandler(event:MouseEvent):void {
            if(textBlock.firstLine)
                removeChild(textBlock.firstLine);
```

```
            var newFormat:ElementFormat = textBlock.content.elementFormat.clone();
            switch(newFormat.color)
            {
                case 0x000000:
                    newFormat.color = 0xFF0000;
                    break;
                case 0xFF0000:
                    newFormat.color = 0x00FF00;
                    break;
                case 0x00FF00:
                    newFormat.color = 0x0000FF;
                    break;
                case 0x0000FF:
                    newFormat.color = 0x000000;
                    break;
            }
            textBlock.content.elementFormat = newFormat;
            createLine(textBlock);
        }
    }
}
```

Copie miroir d'événements

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez également matérialiser les événements d'un bloc de texte (ou d'une portion d'un bloc de texte) dans un diffuseur d'événements. Vous devez tout d'abord créer une occurrence de EventDispatcher, puis l'affecter à la propriété `eventMirror` d'une occurrence de TextElement. Si le bloc de texte n'est constitué que d'un seul élément de texte, le moteur de saisie effectue une copie miroir des événements pour l'intégralité du bloc de texte. Si le bloc de texte est constitué de plusieurs éléments de texte, le moteur de saisie effectue une copie miroir des événements uniquement pour les occurrences de TextElement dont la propriété `eventMirror` est définie. Dans l'exemple suivant, le texte est constitué de trois éléments : le mot « Click », le mot « here » et la chaîne « to see me in italic ». L'exemple affecte un diffuseur d'événements au deuxième élément de texte, le mot « here », et ajoute un écouteur d'événement, la méthode `clickHandler()`. La méthode `clickHandler()` met le texte en italique. Elle modifie également le contenu du troisième élément de texte et le remplace par la chaîne suivante : « Click here to see me in normal font! ».

```
package
{
    import flash.text.engine.*;
    import flash.ui.Mouse;
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.events.EventDispatcher;

    public class EventMirrorExample extends Sprite
    {
        var fontDescription:FontDescription = new FontDescription("Helvetica", "bold");
        var format:ElementFormat = new ElementFormat(fontDescription, 18);
        var textElement1 = new TextElement("Click ", format);
        var textElement2 = new TextElement("here ", format);
        var textElement3 = new TextElement("to see me in italic! ", format);
        var textBlock:TextBlock = new TextBlock();

        public function EventMirrorExample()
        {
            var myEvent:EventDispatcher = new EventDispatcher();

            myEvent.addEventListener("click", clickHandler);
            myEvent.addEventListener("mouseOut", mouseOutHandler);
            myEvent.addEventListener("mouseOver", mouseOverHandler);

            textElement2.eventMirror=myEvent;

            var groupVector:Vector.<ContentElement> = new Vector.<ContentElement>;
            groupVector.push(textElement1, textElement2, textElement3);
            var groupElement:GroupElement = new GroupElement(groupVector);

            textBlock.content = groupElement;
            createLines(textBlock);
        }

        private function clickHandler(event:MouseEvent):void
        {
            var newFont:FontDescription = new FontDescription();
            newFont.fontWeight = "bold";

            var newFormat:ElementFormat = new ElementFormat();
            newFormat.fontSize = 18;
            if(textElement3.text == "to see me in italic! ") {
                newFont.fontPosture = FontPosture.ITALIC;
                textElement3.replaceText(0,21, "to see me in normal font! ");
            }
            else {
                newFont.fontPosture = FontPosture.NORMAL;
                textElement3.replaceText(0, 26, "to see me in italic! ");
            }
            newFormat.fontDescription = newFont;
            textElement1.elementFormat = newFormat;
            textElement2.elementFormat = newFormat;
            textElement3.elementFormat = newFormat;
            createLines(textBlock);
        }
```

```
        private function mouseOverHandler(event:MouseEvent):void
        {
            Mouse.cursor = "button";
        }

        private function mouseOutHandler(event:MouseEvent):void
        {
                Mouse.cursor = "arrow";
        }

        private function createLines(textBlock:TextBlock):void
        {
            if(textBlock.firstLine)
                removeChild (textBlock.firstLine);
            var textLine:TextLine = textBlock.createTextLine (null, 300);
            textLine.x = 15;
            textLine.y = 20;
            addChild (textLine);
        }
    }
}
```

Grâce aux fonctions `mouseOverHandler()` et `mouseOutHandler()`, le curseur prend la forme d'un bouton lorsqu'il est placé sur le mot « here » et reprend la forme d'une flèche lorsqu'il ne l'est pas.

Mise en forme du texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Un objet `TextBlock` est un objet usine pour la création de lignes de texte. Le contenu d'un objet `TextBlock` est affecté via l'objet `TextElement`. Un objet `ElementFormat` gère la mise en forme du texte. La classe ElementFormat définit certaines propriétés, telles que l'alignement sur la ligne de base, le crénage, l'interlettrage, la rotation du texte, la taille et la couleur des polices, ainsi que la casse. Elle comprend également la méthode `FontDescription`, décrite en détail à la section « Utilisation des polices » à la page 422.

Utilisation de l'objet ElementFormat

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le constructeur de l'objet `ElementFormat` prend des nombreux paramètres facultatifs, dont `FontDescription`. Vous pouvez également définir ces propriétés en dehors du constructeur. L'exemple suivant illustre la relation des différents objets lors de la définition et de l'affichage d'une ligne de texte simple :

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class ElementFormatExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef:ElementFormat;
        private var fd:FontDescription = new FontDescription();
        private var str:String;
        private var tl:TextLine;

        public function ElementFormatExample()
        {
            fd.fontName = "Garamond";
            ef = new ElementFormat(fd);
            ef.fontSize = 30;
            ef.color = 0xFF0000;
            str = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            tl = tb.createTextLine(null,600);
            addChild(tl);
        }
    }
}
```

Couleur de police et transparence (alpha)

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `color` de l'objet `ElementFormat` définit la couleur de police. La valeur est un entier représentant les composants RVB de la couleur, par exemple : 0xFF0000 pour le rouge et 0x00FF00 pour le vert. La valeur par défaut est noir (0x000000).

La propriété `alpha` définit la valeur de transparence alpha d'un élément (`TextElement` et `GraphicElement`). La plage des valeurs est comprise entre 0 (complètement transparent) et 1 (complètement opaque), qui est la valeur par défaut. Les éléments dont la propriété `alpha` est de 0 sont invisibles, mais restent actifs. Cette valeur est multipliée par l'une des valeurs alpha héritées, ce qui rend l'élément plus transparent.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

Alignement et décalage de la ligne de base

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La police et la taille du plus grand texte dans une ligne déterminent sa ligne de base dominante. Vous pouvez écraser ces valeurs en définissant `TextBlock.baselineFontDescription` et `TextBlock.baselineFontSize`. Vous pouvez aligner la ligne de base dominante sur l'une des lignes de bases du texte, à savoir la ligne ascendante, la ligne descendante, ou la ligne de base idéographique supérieure, centrale ou inférieure.

***A.** Ascendante **B.** Ligne de base **C.** Descendante **D.** Hauteur-x*

Dans l'objet `ElementFormat`, trois propriétés déterminent la ligne de base et les caractéristiques d'alignement. La propriété `alignmentBaseline` définit la ligne de base principale d'un objet `TextElement` ou `GraphicElement`. Cette ligne de base est la ligne « d'accrochage ».de l'élément, et c'est à cette position que la ligne de base dominante du texte s'aligne.

La propriété `dominantBaseline` indique la ligne de base de l'élément à utiliser, qui détermine la position verticale de l'élément sur la ligne. La valeur par défaut est `TextBaseline.ROMAN`, mais vous pouvez également stipuler que la ligne de base `IDEOGRAPHIC_TOP` ou `IDEOGRAPHIC_BOTTOM` doit être dominante.

La propriété `baselineShift` déplace la ligne de base selon un nombre de pixels défini sur l'axe y. Dans un texte normal (aucune rotation), une valeur positive déplace la ligne de base vers le bas et une valeur négative la déplace vers le haut.

Casse typographique

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `TypographicCase` de l'objet `ElementFormat` spécifie la casse du texte : majuscule, minuscule ou petites capitales.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

Rotation du texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez appliquer une rotation au bloc de texte ou aux glyphes d'un segment de texte par incréments de 90°. La classe TextRotation définit les constantes suivantes pour définir la rotation du bloc de texte et des glyphes :

Constante	Valeur	Description
AUTO	"auto"	Spécifie une rotation vers la gauche de 90 degrés. Généralement utilisée avec un texte asiatique vertical pour faire pivoter uniquement les glyphes auxquelles il est nécessaire d'appliquer une rotation.
ROTATE_0	"rotate_0"	Spécifie aucune rotation.
ROTATE_180	"rotate_180"	Spécifie une rotation de 180 degrés.
ROTATE_270	"rotate_270"	Spécifie une rotation de 270 degrés.
ROTATE_90	"rotate_90"	Spécifie une rotation vers la droite de 90 degrés.

Pour appliquer une rotation aux lignes d'un texte, définissez tout d'abord la propriété `TextBlock.lineRotation` avant d'appeler la méthode `TextBlock.createTextLine()` pour créer la ligne de texte.

Pour appliquer une rotation aux glyphes dans un bloc de texte ou un segment, définissez la propriété `ElementFormat.textRotation` sur le nombre de degrés de rotation des glyphes souhaité. Une glyphe est la forme qui constitue un caractère, ou une partie d'un caractère qui consiste en plusieurs glyphes. La lettre « a » et le point sur un « i », par exemple, sont des glyphes.

La rotation des glyphes est importante dans certaines langues asiatiques, notamment si vous souhaitez appliquer une rotation verticale aux lignes, mais ne pas appliquer de rotation aux caractères dans les lignes. Pour plus d'informations sur la rotation du texte asiatique, voir « Justification du texte asiatique » à la page 426.

Voici un exemple de rotation du texte et des glyphes qu'il contient dans un texte asiatique : Cet exemple utilise également une police japonaise :

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class RotationExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef:ElementFormat;
        private var fd:FontDescription = new FontDescription();
        private var str:String;
        private var tl:TextLine;

        public function RotationExample()
        {
            fd.fontName = "MS Mincho";
            ef = new ElementFormat(fd);
            ef.textRotation = TextRotation.AUTO;
            str = "This is rotated Japanese text";
            te = new TextElement(str, ef);
            tb.lineRotation = TextRotation.ROTATE_90;
            tb.content = te;
            tl = tb.createTextLine(null,600);
            addChild(tl);
        }
    }
}
```

Verrouillage et clonage d'un objet ElementFormat

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Lorsqu'un objet `ElementFormat` est affecté à un type de `ContentElement`, sa propriété `locked` est automatiquement définie sur `true`. Toute tentative de modification d'un objet `ElementFormat` verrouillé renvoie une erreur `IllegalOperationError`. La meilleure pratique consiste à définir complètement ce type d'objet avant de l'affecter à l'occurrence de `TextElement`.

Si vous souhaitez modifier une occurrence de `ElementFormat` existante, vous devez tout d'abord vérifier sa propriété `locked`. Si elle est définie sur `true`, utilisez la méthode `clone()` pour créer une copie déverrouillée de l'objet. Vous pouvez modifier les propriétés de cet objet déverrouillé, puis l'affecter à l'occurrence de `TextElement`. Toute nouvelle ligne créée à partir de cet objet adopte la nouvelle mise en forme. Les lignes précédentes créées à partir de cet objet et utilisant l'ancienne mise en forme ne sont pas modifiées.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class ElementFormatCloneExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var ef2:ElementFormat;
        private var fd:FontDescription = new FontDescription();

        public function ElementFormatCloneExample()
        {
            fd.fontName = "Garamond";
            ef1 = new ElementFormat(fd);
            ef1.fontSize = 24;
            var str:String = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            var tx1:TextLine = tb.createTextLine(null,600);
            addChild(tx1);

            ef2 = (ef1.locked) ? ef1.clone() : ef1;
            ef2.fontSize = 32;
            tb.content.elementFormat = ef2;
            var tx2:TextLine = tb.createTextLine(null,600);
            addChild(tx2);
        }
    }
}
```

Utilisation des polices

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'objet `FontDescription` est utilisé avec l'occurrence de `ElementFormat` pour identifier une police et définir certaines de ses caractéristiques. Ces caractéristiques incluent le nom de la police, le poids, la position, le rendu et la méthode de recherche de la police (police de périphérique/police intégrée).

Remarque : *FTE ne prend pas en charge les polices Type 1 ou les polices bitmap, telles que Type 3, ATC, CID ou CID basées sur SFNT.*

Définition des caractéristiques des polices (objet FontDescription)

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `fontName` de l'objet `FontDescription` peut être un nom unique ou une liste de noms séparés par des virgules. Par exemple, dans une liste telle que « Arial, Helvetica, _sans », Text Engine recherche tout d'abord « Arial », puis « Helvetica » et finalement « _sans » s'il ne parvient pas à trouver les deux premières polices. La définition des noms de police comprend trois noms de police génériques : « _sans », « _serif » et « _typewriter ». Ces noms correspondent à des polices de périphérique spécifiques, selon le système de lecture. Il est judicieux de spécifier ce type de noms par défaut dans toutes les descriptions de police qui utilisent des polices de périphérique. Si la propriété `fontName` n'est pas spécifiée, « _serif » est utilisé comme nom par défaut.

La propriété `fontPosture` peut être définie sur la valeur par défaut (`FontPosture.NORMAL`) ou en italique (`FontPosture.ITALIC`). La propriété `fontWeight` peut être définie sur la valeur par défaut (`FontWeight.NORMAL`) ou en caractères gras (`FontWeight.BOLD`).

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

Polices intégrées ou polices de périphérique ?

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `fontLookup` de l'objet `FontDescription` indique si Text Engine recherche une police de périphérique ou une police intégrée pour rendre le texte. Si une police de périphérique (`FontLookup.DEVICE`) est spécifiée, le moteur d'exécution recherche la police sur le système de lecture. Si vous définissez une police intégrée (`FontLookup.EMBEDDED_CFF`), le moteur d'exécution recherche une police de ce type portant le nom indiqué dans le fichier SWF. Seules les polices CFF (Compact Font Format) intégrées utilisent ce paramètre. Si la police spécifiée est introuvable, une police de périphérique est utilisée.

Les polices de périphérique donnent lieu à des fichiers SWF moins volumineux. Les polices intégrées garantissent une plus grande homogénéité d'une plate-forme à l'autre.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

Mode de rendu et repères

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le rendu CFF (Compact Font Format) est disponible à partir de Flash Player 10 et Adobe AIR 1.5. Ce type de rendu de police permet une meilleure lisibilité du texte et un affichage optimisé des caractères de petite taille. Ce paramètre s'applique uniquement aux polices intégrées. La valeur par défaut de `FontDescription` correspond à ce paramètre (`RenderingMode.CFF`) pour la propriété `renderingMode`. Vous pouvez définir cette propriété sur `RenderingMode.NORMAL` afin qu'elle corresponde au type de rendu utilisé par Flash Player 7 ou versions antérieures.

Lorsque le rendu CFF est sélectionné, une deuxième propriété, `cffHinting`, permet de contrôler la manière dont les corps horizontaux d'une police sont adaptés à la grille de sous-pixels. La valeur par défaut, `CFFHinting.HORIZONTAL_STEM`, utilise les repères CFF. Si vous définissez cette propriété sur `CFFHinting.NONE` , les repères sont supprimés. Ce paramètre convient pour les animations ou les grandes tailles de police.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

Verrouillage et clonage d'un objet FontDescription

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Lorsque vous affectez une occurrence de `ElementFormat` à un objet `FontDescription`, la propriété `locked` de ce dernier est automatiquement définie sur `true`. Toute tentative de modification d'un objet `FontDescription` verrouillé renvoie une erreur `IllegalOperationError`. La meilleure pratique consiste à définir complètement ce type d'objet avant de l'affecter à l'occurrence de `ElementFormat`.

Si vous souhaitez modifier un objet `FontDescription` existant, vous devez tout d'abord vérifier sa propriété `locked`. Si elle est définie sur `true`, utilisez la méthode `clone()` pour créer une copie déverrouillée de l'objet. Vous pouvez modifier les propriétés de cet objet déverrouillé, puis l'affecter à `ElementFormat`. Toute nouvelle ligne créée à partir de ce `TextElement` adopte la nouvelle mise en forme. Les lignes précédentes créées à partir de ce même objet restent inchangées.

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class FontDescriptionCloneExample extends Sprite
    {
        private var tb:TextBlock = new TextBlock();
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var ef2:ElementFormat;
        private var fd1:FontDescription = new FontDescription();
        private var fd2:FontDescription;

        public function FontDescriptionCloneExample()
        {
            fd1.fontName = "Garamond";
            ef1 = new ElementFormat(fd);
            var str:String = "This is flash text";
            te = new TextElement(str, ef);
            tb.content = te;
            var tx1:TextLine = tb.createTextLine(null,600);
            addChild(tx1);

            fd2 = (fd1.locked) ? fd1.clone() : fd1;
            fd2.fontName = "Arial";
            ef2 = (ef1.locked) ? ef1.clone() : ef1;
            ef2.fontDescription = fd2;
            tb.content.elementFormat = ef2;
            var tx2:TextLine = tb.createTextLine(null,600);
            addChild(tx2);
        }
    }
}
```

Contrôle du texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

FTE vous propose un ensemble de commandes de mise en forme du texte afin de gérer la justification et l'espacement des caractères (crénage et interlettrage). Il existe également des propriétés permettant de détecter les lignes brisées et de définir des taquets de tabulation dans les lignes.

Justification du texte

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

En ajustant l'espacement entre les mots, et parfois entre les lettres, toutes les lignes d'un paragraphe possèdent une longueur identique. Bien que l'espacement entre les mots et les lettres varie, le texte est aligné des deux côtés. Les colonnes de texte dans les journaux et les magazines sont le plus souvent justifiées.

La propriété `lineJustfication` de la classe SpaceJustifier vous permet de contrôler la justification des lignes dans un bloc de texte. La classe LineJustification définit des constantes en vue de spécifier une option de justification : `ALL_BUT_LAST` justifie tout le texte, à l'exception de la dernière ligne ; `ALL_INCLUDING_LAST` justifie tout le texte, y compris la dernière ligne ; l'option par défaut, `UNJUSTIFIED`, ne justifie pas le texte.

Pour justifier le texte, définissez la propriété `lineJustification` sur une occurrence de la classe SpaceJustifier, puis affectez celle-ci à la propriété `textJustifier` d'une occurrence de TextBlock. L'exemple suivant crée un paragraphe dans lequel la totalité du texte est justifiée, à l'exception de la dernière ligne.

```
package
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class JustifyExample extends Sprite
    {
        public function JustifyExample()
        {
            var str:String = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, " +
            "sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut " +
            "enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut " +
            "aliquip ex ea commodo consequat.";

            var format:ElementFormat = new ElementFormat();
            var textElement:TextElement=new TextElement(str,format);
            var spaceJustifier:SpaceJustifier=new
SpaceJustifier("en",LineJustification.ALL_BUT_LAST);

            var textBlock:TextBlock = new TextBlock();
            textBlock.content=textElement;
            textBlock.textJustifier=spaceJustifier;
            createLines(textBlock);
        }

        private function createLines(textBlock:TextBlock):void {
            var yPos=20;
            var textLine:TextLine=textBlock.createTextLine(null,150);

            while (textLine) {
                addChild(textLine);
                textLine.x=15;
                yPos+=textLine.textHeight+2;
                textLine.y=yPos;
                textLine=textBlock.createTextLine(textLine,150);
            }
        }
    }
}
```

Pour varier l'espacement entre les lettres et entre les mots, définissez la propriété `SpaceJustifier.letterspacing` sur `true`. L'activation de l'espacement entre les lettres peut réduire le nombre d'espaces disgracieux entre les mots, qui se produisent parfois lors d'une simple justification.

Justification du texte asiatique

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Pour justifier un texte asiatique, il faut tenir compte d'autres considérations. Le texte peut être écrit de haut en haut et certains caractères appelés kinsoku ne peuvent pas apparaître au début ou à la fin d'une ligne. La classe JustificationStyle définit les constantes suivantes, qui spécifient les options de gestion de ces caractères. `PRIORITIZE_LEAST_ADJUSTMENT` base la justification sur l'expansion ou sur la compression de la ligne, en fonction de celle qui produit les meilleurs résultats. `PUSH_IN_KINSOKU` base la justification sur la compression des caractères kinsoku à la fin de la ligne, ou sur l'expansion de la ligne s'il n'existe aucun caractère kinsoku ou si cet espace est insuffisant.

PUSH_OUT_ONLY base la justification sur l'expansion de la ligne. Pour créer un bloc de texte asiatique vertical, définissez la propriété TextBlock.lineRotation sur TextRotation.ROTATE_90, puis définissez la propriété ElementFormat.textRotation sur TextRotation.AUTO (paramètre par défaut). Si vous définissez la propriété textRotation sur AUTO, les glyphes dans le texte restent verticales au lieu de pivoter sur le côté lors de la rotation de la ligne. Le paramètre AUTO effectue une rotation vers la gauche de 90° pour les glyphes complètes uniquement, comme le spécifient les propriétés Unicode de la glyphe. L'exemple suivant affiche un bloc de texte japonais vertical et le justifie à l'aide de l'option PUSH_IN_KINSOKU.

```
package
{
    import flash.text.engine.*;
    import flash.display.Stage;
    import flash.display.Sprite;
    import flash.system.Capabilities;

    public class EastAsianJustifyExample extends Sprite
    {
        public function EastAsianJustifyExample()
        {
            var Japanese_txt:String = String.fromCharCode(
            0x5185, 0x95A3, 0x5E9C, 0x304C, 0x300C, 0x653F, 0x5E9C, 0x30A4,
            0x30F3, 0x30BF, 0x30FC, 0x30CD, 0x30C3, 0x30C8, 0x30C6, 0x30EC,
            0x30D3, 0x300D, 0x306E, 0x52D5, 0x753B, 0x914D, 0x4FE1, 0x5411,
            0x3051, 0x306B, 0x30A2, 0x30C9, 0x30D3, 0x30B7, 0x30B9, 0x30C6,
            0x30E0, 0x30BA, 0x793E, 0x306E)
            var textBlock:TextBlock = new TextBlock();
            var font:FontDescription = new FontDescription();
            var format:ElementFormat = new ElementFormat();
            format.fontSize = 12;
            format.color = 0xCC0000;
            format.textRotation = TextRotation.AUTO;
            textBlock.baselineZero = TextBaseline.IDEOGRAPHIC_CENTER;
            var eastAsianJustifier:EastAsianJustifier = new EastAsianJustifier("ja",
LineJustification.ALL_BUT_LAST);
            eastAsianJustifier.justificationStyle = JustificationStyle.PUSH_IN_KINSOKU;
            textBlock.textJustifier = eastAsianJustifier;
            textBlock.lineRotation = TextRotation.ROTATE_90;
            var linePosition:Number = this.stage.stageWidth - 75;
            if (Capabilities.os.search("Mac OS") > -1)
                // set fontName: Kozuka Mincho Pro R
```

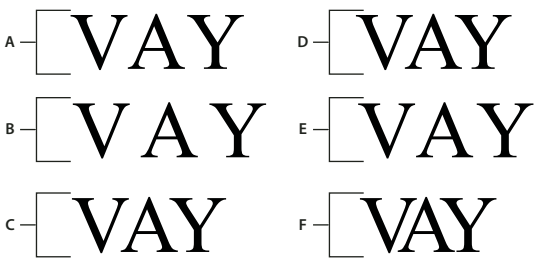
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            while (true)
            {
                var textLine:TextLine = textBlock.createTextLine(previousLine, 200);
                if (textLine == null)
                    break;
                textLine.y = 20;
                textLine.x = linePosition;
                linePosition -= 25;
                addChild(textLine);
                previousLine = textLine;
            }
        }
    }
}
```

Crénage et interlettrage

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le crénage et l'interlettrage influent sur la distance entre les paires de caractères adjacentes dans un bloc de texte. Le crénage contrôle la manière dont les paires de caractères s'assemblent, notamment les paires « WA » ou « Va ». Il est défini dans l'objet `ElementFormat`. Par défaut, cet effet est activé (`Kerning.ON`) ; il peut être réglé sur OFF ou AUTO, auquel cas le crénage n'est appliqué qu'entre les caractères autres que Kanji, Hiragana ou Katakana.

L'interlettrage ajoute ou soustrait un nombre de pixels défini entre les caractères dans un bloc de texte ; cet effet est également défini dans l'objet `ElementFormat`. L'interlettrage fonctionne avec les polices intégrées et les polices de périphérique. FTE prend en charge deux propriétés d'interlettrage : `trackingLeft`, qui ajoute ou soustrait des pixels à gauche d'un caractère, et `trackingRight`, qui ajoute ou soustrait des pixels à droite d'un caractère. Si vous utilisez le crénage, la valeur d'interlettrage est ajoutée aux valeurs de crénage ou soustraite des valeurs de crénage de chaque paire de caractères.



***A.** Kerning.OFF **B.** TrackingRight=5, Kerning.OFF **C.** TrackingRight=-5, Kerning.OFF **D.** Kerning.ON **E.** TrackingRight=-5, Kerning.ON **F.** TrackingRight=-5, Kerning.ON*

```
var ef1:ElementFormat = new ElementFormat();
ef1.kerning = Kerning.OFF;

var ef2:ElementFormat = new ElementFormat();
ef2.kerning = Kerning.ON;
ef2.trackingLeft = 0.8;
ef2.trackingRight = 0.8;

var ef3:ElementFormat = new ElementFormat();
ef3.trackingRight = -0.2;
```

Texte avec retour à la ligne automatique

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La propriété `breakOpportunity` de l'objet `ElementFormat` indique les caractères pouvant être utilisés pour le renvoi lorsque le texte est renvoyé sur plusieurs lignes. La valeur par défaut, `BreakOpportunity.AUTO`, utilise les propriétés Unicode standard, telles que le saut entre les mots et sur les traits d'union. Le paramètre `BreakOpportunity.ALL` permet de considérer un caractère comme une opportunité de saut de ligne, ce qui est très utile lors de la création d'effets, tels que le texte le long d'un tracé.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

Taquets de tabulation

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Pour définir des taquets de tabulation dans un bloc de texte, définissez les taquets de tabulation en créant des occurrences de la classe TabStop. Les paramètres du constructeur `TabStop()` indiquent la méthode d'alignement du texte avec le taquet de tabulation. Ces paramètres indiquent la position du taquet de tabulation, et pour l'alignement décimal, la valeur d'alignement, exprimée sous forme de chaîne. En général, cette valeur est un point décimal, mais elle pourrait également être une virgule ou le symbole du dollar, du yen ou de l'euro, par exemple. La ligne de code suivante crée un taquet de tabulation appelé tab1.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

Après avoir créé les taquets de tabulation d'un bloc de texte, affectez-les à la propriété `tabStops` d'une occurrence de TextBlock. Etant donné que la propriété `tabStops` nécessite un vecteur, créez tout d'abord un vecteur, puis ajoutez la tabulation pour l'arrêter. Le vecteur vous permet d'affecter un ensemble de taquets de tabulation au bloc de texte. L'exemple suivant crée une occurrence de `Vector<TabStop>` et lui ajoute un ensemble d'objets TabStop. Affectez ensuite les taquets de tabulation à la propriété `tabStops` d'une occurrence de TextBlock.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Pour plus d'informations sur les vecteurs, voir « Utilisation de tableaux » à la page 25.

L'exemple suivant illustre l'effet de chacune des options d'alignement de l'objet TabStop.

```
package {

    import flash.text.engine.*;
    import flash.display.Sprite;

    public class TabStopExample extends Sprite
    {
        public function TabStopExample()
        {
            var format:ElementFormat = new ElementFormat();
            format.fontDescription = new FontDescription("Arial");
            format.fontSize = 16;

            var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
            tabStops.push(
                new TabStop(TabAlignment.START, 20),
                new TabStop(TabAlignment.CENTER, 140),
                new TabStop(TabAlignment.DECIMAL, 260, "."),
                new TabStop(TabAlignment.END, 380));
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = new TextElement(
                "\tt1\tt2\tt3\tt4\n" +
                "\tThis line aligns on 1st tab\n" +
                "\t\t\t\tThis is the end\n" +
                "\tThe following fragment centers on the 2nd tab:\t\t\n" +
                "\t\tit's on me\t\t\n" +
                "\tThe following amounts align on the decimal point:\n" +
                "\t\t\t45.00\t\n" +
                "\t\t\t75,320.00\t\n" +
                "\t\t\t6,950.00\t\n" +
                "\t\t\t7.01\t\n", format);

            textBlock.tabStops = tabStops;
            var yPosition:Number = 60;
            var previousTextLine:TextLine = null;
            var textLine:TextLine;
            var i:int;
            for (i = 0; i < 10; i++) {
                textLine = textBlock.createTextLine(previousTextLine, 1000, 0);
                textLine.x = 20;
                textLine.y = yPosition;
                addChild(textLine);
                yPosition += 25;
                previousTextLine = textLine;
            }
        }
    }
}
```

Exemple d'utilisation de Flash Text Engine : mise en forme d'un article de journal

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Cet exemple de code illustre l'utilisation de FTE (Flash Text Engine) pour mettre en forme une page de journal simple. Cette page comprend un gros titre, un sous-titre et une section sur plusieurs colonnes.

Créez d'abord un fichier FLA et rattachez le code suivant au cadre n° 2 de la couche par défaut :

```
import com.example.programmingas3.newslayout.StoryLayout ;
// frame sc ript - create  a 3-columned arti cle layout
var story:StoryLayout = new StoryLayout(720, 500, 3, 10);
story.x = 20;
story.y = 80;
addChild(story);
stop();
```

Dans cet exemple, StoryLayout.as est le script contrôleur. Il définit le contenu, lit les informations de style issues d'une feuille de style externe et les affecte aux objets ElementFormat. Il crée ensuite le gros titre, le sous-titre et les éléments de texte sur plusieurs colonnes.

```
package com.example.programmingas3.newslayout
{
    import flash.display.Sprite;
    import flash.text.StyleSheet;
    import flash.text.engine.*;

    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.net.URLLoader;
    import flash.display.Sprite;
    import flash.display.Graphics;

    public class StoryLayout extends Sprite
    {
        public var headlineTxt:HeadlineTextField;
        public var subtitleTxt:HeadlineTextField;
        public var storyTxt:MultiColumnText;
        public var sheet:StyleSheet;
        public var h1_ElFormat:ElementFormat;
        public var h2_ElFormat:ElementFormat;
        public var p_ElFormat:ElementFormat;

        private var loader:URLLoader;

        public var paddingLeft:Number;
        public var paddingRight:Number;
        public var paddingTop:Number;
        public var paddingBottom:Number;

        public var preferredWidth:Number;
        public var preferredHeight:Number;

        public var numColumns:int;
```

```
        public var bgColor:Number = 0xFFFFFF;

        public var headline:String = "News Layout Example";
        public var subtitle:String = "This example formats text like a newspaper page using the
Flash Text Engine API. ";

        public var rawTestData:String =
        "From the part Mr. Burke took in the American Revolution, it was natural that I should
consider him a friend to mankind; and as our acquaintance commenced on that ground, it would
have been more agreeable to me to have had cause to continue in that opinion than to change it. " +
        "At the time Mr. Burke made his violent speech last winter in the English Parliament
against the French Revolution and the National Assembly, I was in Paris, and had written to him
but a short time before to inform him how prosperously matters were going on. Soon after this I
saw his advertisement of the Pamphlet he intended to publish: As the attack was to be made in a
language but little studied, and less understood in France, and as everything suffers by
translation, I promised some of the friends of the Revolution in that country that whenever Mr.
Burke's Pamphlet came forth, I would answer it. This appeared to me the more necessary to be
done, when I saw the flagrant misrepresentations which Mr. Burke's Pamphlet contains; and that
while it is an outrageous abuse on the French Revolution, and the principles of Liberty, it is
an imposition on the rest of the world. " +
        "I am the more astonished and disappointed at this conduct in Mr. Burke, as (from the
circumstances I am going to mention) I had formed other expectations. " +
        "I had seen enough of the miseries of war, to wish it might never more have existence
in the world, and that some other mode might be found out to settle the differences that should
occasionally arise in the neighbourhood of nations. This certainly might be done if Courts were
disposed to set honesty about it, or if countries were enlightened enough not to be made the
dupes of Courts. The people of America had been bred up in the same prejudices against France,
which at that time characterised the people of England; but experience and an acquaintance with
the French Nation have most effectually shown to the Americans the falsehood of those prejudices;
and I do not believe that a more cordial and confidential intercourse exists between any two
countries than between America and France. ";

        public function StoryLayout(w:int = 400, h:int = 200, cols:int = 3, padding:int =
10):void
        {
            this.preferredWidth = w;
            this.preferredHeight = h;

            this.numColumns = cols;

            this.paddingLeft = padding;
            this.paddingRight = padding;
            this.paddingTop = padding;
            this.paddingBottom = padding;

            var req:URLRequest = new URLRequest("story.css");
            loader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, onCSSFileLoaded);
            loader.load(req);
        }

        public function onCSSFileLoaded(event:Event):void
        {
            this.sheet = new StyleSheet();
            this.sheet.parseCSS(loader.data);

            // convert headline styles to ElementFormat objects
```

```
    h1_ElFormat = getElFormat("h1", this.sheet);
    h1_ElFormat.typographicCase = TypographicCase.UPPERCASE;
    h2_ElFormat = getElFormat("h2", this.sheet);
    p_ElFormat = getElFormat("p", this.sheet);
    displayText();
}

public function drawBackground():void
{
    var h:Number = this.storyTxt.y + this.storyTxt.height +
                    this.paddingTop + this.paddingBottom;
    var g:Graphics = this.graphics;
    g.beginFill(this.bgColor);
    g.drawRect(0, 0, this.width + this.paddingRight + this.paddingLeft, h);
    g.endFill();
}

/**
 * Reads a set of style properties for a named style and then creates
 * a TextFormat object that uses the same properties.
 */
public function getElFormat(styleName:String, ss:StyleSheet):ElementFormat
{
var style:Object = ss.getStyle(styleName);
if (style != null)
{
var colorStr:String = style.color;
if (colorStr != null && colorStr.indexOf("#") == 0)
{
    style.color = colorStr.substr(1);
}
        var fd:FontDescription = new FontDescription(
                              style.fontFamily,
                              style.fontWeight,
                              FontPosture.NORMAL,
                              FontLookup.DEVICE,
                              RenderingMode.NORMAL,
                              CFFHinting.NONE);
var format:ElementFormat = new ElementFormat(fd,
                       style.fontSize,
                       style.color,
                       1,
                       TextRotation.AUTO,
                       TextBaseline.ROMAN,
                       TextBaseline.USE_DOMINANT_BASELINE,
                       0.0,
                       Kerning.ON,
                       0.0,
                       0.0,
                       "en",
                               BreakOpportunity.AUTO,
                               DigitCase.DEFAULT,
                               DigitWidth.DEFAULT,
                       LigatureLevel.NONE,
                               TypographicCase.DEFAULT);

if (style.hasOwnProperty("letterSpacing"))
```

```
        {
            format.trackingRight = style.letterSpacing;
        }
        }
        return format;
        }

        public function displayText():void
        {
        headlineTxt = new HeadlineTextField(h1_ElFormat,headline,this.preferredWidth);
        headlineTxt.x = this.paddingLeft;
        headlineTxt.y = 40 + this.paddingTop;
            headlineTxt.fitText(1);
            this.addChild(headlineTxt);

        subtitleTxt = new HeadlineTextField(h2_ElFormat,subtitle,this.preferredWidth);
        subtitleTxt.x = this.paddingLeft;
        subtitleTxt.y = headlineTxt.y + headlineTxt.height;
            subtitleTxt.fitText(2);
        this.addChild(subtitleTxt);

        storyTxt = new MultiColumnText(rawTestData, this.numColumns,
                        20, this.preferredWidth, this.preferredHeight, p_ElFormat);
        storyTxt.x = this.paddingLeft;
        storyTxt.y = subtitleTxt.y + subtitleTxt.height + 10;
        this.addChild(storyTxt);

            drawBackground();
        }
    }
}
```

FormattedTextBlock.as est utilisé en tant que classe de base pour la création des blocs de texte. Il comprend également des fonctions permettant de modifier la taille de police et la casse.

```
package com.example.programmingas3.newslayout
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class FormattedTextBlock extends Sprite
    {
        public var tb:TextBlock;
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var textWidth:int;
        public var totalTextLines:int;
        public var blockText:String;
        public var leading:Number = 1.25;
        public var preferredWidth:Number = 720;
        public var preferredHeight:Number = 100;

        public function FormattedTextBlock(ef:ElementFormat,txt:String, colW:int = 0)
        {
            this.textWidth = (colW==0) ? preferredWidth : colW;
            blockText = txt;
            ef1 = ef;
```

```
        tb = new TextBlock();
        tb.textJustifier = new SpaceJustifier("en",LineJustification.UNJUSTIFIED,false);
        te = new TextElement(blockText,this.ef1);
        tb.content = te;
        this.breakLines();
    }

    private function breakLines()
    {
        var textLine:TextLine = null;
        var y:Number = 0;
        var lineNum:int = 0;
        while (textLine = tb.createTextLine(textLine,this.textWidth,0,true))
        {
            textLine.x = 0;
            textLine.y = y;
            y += this.leading*textLine.height;
            this.addChild(textLine);
        }
        for (var i:int = 0; i < this.numChildren; i++)
        {
            TextLine(this.getChildAt(i)).validity = TextLineValidity.STATIC;
        }
        this.totalTextLines = this.numChildren;
    }

    private function rebreakLines()
    {
        this.clearLines();
        this.breakLines();
    }

    private function clearLines()
    {
        while(this.numChildren)
        {
            this.removeChildAt(0);
        }
    }
```

```
        public function changeSize(size:uint=12):void
        {
            if (size > 5)
            {
                var ef2:ElementFormat = ef1.clone();
                ef2.fontSize = size;
                te.elementFormat = ef2;
                this.rebreakLines();
            }
        }

        public function changeCase(newCase:String = "default"):void
        {
            var ef2:ElementFormat = ef1.clone();
            ef2.typographicCase = newCase;
            te.elementFormat = ef2;
        }
    }
}
```

HeadlineTextBlock.as étend la classe FormattedTextBlock et est utilisé pour créer les gros titres. Il contient une fonction destinée à faire tenir le texte dans un espace déterminé sur la page.

```
package com.example.programmingas3.newslayout
{
    import flash.text.engine.*;
    public class HeadlineTextField extends FormattedTextBlock
    {

        public static var MIN_POINT_SIZE:uint = 6;
        public static var MAX_POINT_SIZE:uint = 128;

        public function HeadlineTextField(te:ElementFormat,txt:String,colW:int = 0)
        {
            super(te,txt);
        }

        public function fitText(maxLines:uint = 1, targetWidth:Number = -1):uint
        {
            if (targetWidth == -1)
            {
                targetWidth = this.width;
            }

            var pixelsPerChar:Number = targetWidth / this.blockText.length;
            var pointSize:Number = Math.min(MAX_POINT_SIZE,
                           Math.round(pixelsPerChar * 1.8 * maxLines));

            if (pointSize < 6)
            {
                // the point size is too small
                return pointSize;
            }

            this.changeSize(pointSize);
            if (this.totalTextLines > maxLines)
```

```
            {
                return shrinkText(--pointSize, maxLines);
            }
            else
            {
                return growText(pointSize, maxLines);
            }
        }

        public function growText(pointSize:Number, maxLines:uint = 1):Number
        {
            if (pointSize >= MAX_POINT_SIZE)
            {
                return pointSize;
            }

            this.changeSize(pointSize + 1);
            if (this.totalTextLines > maxLines)
            {
                // set it back to the last size
                this.changeSize(pointSize);
                return pointSize;
            }
            else
            {
                return growText(pointSize + 1, maxLines);
            }
        }

        public function shrinkText(pointSize:Number, maxLines:uint=1):Number
        {
            if (pointSize <= MIN_POINT_SIZE)
            {
                return pointSize;
            }
            this.changeSize(pointSize);

            if (this.totalTextLines > maxLines)
            {
                return shrinkText(pointSize - 1, maxLines);
            }
            else
            {
                return pointSize;
            }
        }
    }
}
```

MultiColumnText.as gère la mise en forme du texte dans un format multicolonne. Il illustre la souplesse d'un objet TextBlock, utilisé comme usine pour créer, mettre en forme et positionner les lignes de texte.

```
package com.example.programmingas3.newslayout
{
    import flash.display.Sprite;
    import flash.text.engine.*;

    public class MultiColumnText extends Sprite
    {
        private var tb:TextBlock;
        private var te:TextElement;
        private var numColumns:uint = 2;
        private var gutter:uint = 10;
        private var leading:Number = 1.25;
        private var preferredWidth:Number = 400;
        private var preferredHeight:Number = 100;
        private var colWidth:int = 200;

        public function MultiColumnText(txt:String = "",cols:uint = 2,
            gutter:uint = 10, w:Number = 400, h:Number = 100,
            ef:ElementFormat = null):void
        {
            this.numColumns = Math.max(1, cols);
            this.gutter = Math.max(1, gutter);

            this.preferredWidth = w;
            this.preferredHeight = h;

            this.setColumnWidth();

            var field:FormattedTextBlock = new FormattedTextBlock(ef,txt,this.colWidth);
            var totLines:int = field.totalTextLines;
            field = null;
            var linesPerCol:int = Math.ceil(totLines/cols);

            tb = new TextBlock();
            te = new TextElement(txt,ef);
            tb.content = te;
            var textLine:TextLine = null;
            var x:Number = 0;
            var y:Number = 0;
            var i:int = 0;
            var j:int = 0;
            while (textLine = tb.createTextLine(textLine,this.colWidth,0,true))
            {
                textLine.x = Math.floor(i/(linesPerCol+1))*(this.colWidth+this.gutter);
                textLine.y = y;
                y += this.leading*textLine.height;
```

```
                j++;
                if(j>linesPerCol)
                {
                    y = 0;
                    j = 0;
                }
                i++;

                this.addChild(textLine);
            }
        }

        private function setColumnWidth():void
        {
        this.colWidth = Math.floor( (this.preferredWidth -
            ((this.numColumns - 1) * this.gutter)) / this.numColumns);
        }

    }
}
```

Chapitre 23 : Utilisation de Text Layout Framework

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Présentation de Text Layout Framework

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Text Layout Framework (TLF) est une bibliothèque ActionScript évolutive. TLF est basé sur le moteur texte d'Adobe® Flash® Player 10 et d'Adobe® AIR® 1.5. Il propose des fonctions typographiques et de mise en forme du texte avancées qui assurent une typographie innovatrice sur le Web. TLF peut être utilisé avec Adobe® Flex® ou Adobe® Flash® Professional. Les développeurs peuvent utiliser ou étendre des composants existants ou faire appel à la structure pour créer leurs propres composants texte.

TLF intègre les fonctionnalités suivantes :

- Texte bidirectionnel, texte vertical et plus de 30 scripts tels que l'arabe, le chinois, le coréen, l'hébreu, le japonais, le laotien, le thaï, le vietnamien, etc.
- Sélection, modification et distribution du texte sur plusieurs colonnes et conteneurs liés
- Texte vertical, Tate-Chu-Yoko (texte horizontal placé entre du texte vertical) et justificateur pour la typographie asiatique
- Contrôles typographiques riches, tels que le crénage, les ligatures, la casse typographique, la casse des chiffres, la largeur des chiffres et les tirets conditionnels
- Couper, copier, coller, annulation et mouvements de souris et clavier standard de modification
- API riches de développement destinées à manipuler le contenu, la mise en forme et le balisage de texte, ainsi qu'à créer des composants texte personnalisés
- Excellente prise en charge des listes, notamment les marques et formats de numérotation personnalisés
- Règles de positionnement et images intégrées

Text Layout Framework est une bibliothèque ActionScript 3.0 basée sur la version de Flash Text Engine (FTE) introduite dans Flash Player 10. Vous pouvez accéder à FTE via le package `flash.text.engine`, qui fait partie intégrante de l'API (Application Programming Interface) de Flash Player 10.

Toutefois, l'API de Flash Player fournit un accès de bas niveau à FTE, ce qui signifie que certaines tâches nécessitent parfois un volume de code relativement important. TLF encapsule le code de bas niveau dans des API simplifiées. Il propose également une architecture conceptuelle qui organise les blocs de construction de base définis par FTE pour former un système convivial.

Contrairement à FTE, TLF n'est pas intégré à Flash Player. Il correspond à une bibliothèque de composants indépendants écrits entièrement en ActionScript 3.0. En raison de l'extensibilité de la structure, il peut être adapté à des environnements déterminés. Flash Professional et le kit de développement (SDK) de Flex contiennent tous deux des composants basés sur la structure TLF.

Voir aussi

Application de marquage TLF "Flow"

Prise en charge des scripts complexes

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

TLF prend en charge les scripts complexes, qui gèrent des fonctionnalités telles que la capacité d'afficher et de modifier les scripts rédigés de droite à gauche. TLF permet également d'afficher et de modifier un mélange de scripts écrits de gauche à droite et de droite à gauche, comme l'arabe et l'hébreu. TLF prend en charge non seulement la mise en forme du texte vertical chinois, coréen et japonais, mais également tate-chu-yoko (éléments TCY). Les éléments TCY sont des blocs de texte horizontaux intégrés à des segments verticaux de texte. Les scripts suivants sont pris en charge :

- Latin (anglais, espagnol, français, vietnamien, etc.)
- Arménien, cyrillique, éthiopien, géorgien et grec
- Arabe et hébreu
- Idéogrammes Han, Kana (chinois, coréen et japonais) et Hangul Johab (coréen)
- Thaï, khmer et laotien
- Bengali, dévanâgarî, gujarâtî, gourmoukhî, kannara, malayalam, oriya, tamoul, télougou et tibétain
- Bouhid, chérokie, deseret, écriture syllabique canadienne, hanounoo, shavian, tagalog, tagbanoua, tifinaghe, vai et yi

Utilisation de Text Layout Framework (TLF) dans Flash Professional et Flex

Vous pouvez créer des composants personnalisés dans Flash directement à partir de classes TLF. Flash Professional CS5 intègre en outre une nouvelle classe, fl.text.TLFTextField, qui encapsule la fonctionnalité TLF. La classe TLFTextField permet de créer des champs de texte en ActionScript qui utilisent les fonctions d'affichage de texte avancées de TLF. Créez un objet TLFTextField à l'instar d'un champ de texte par le biais de la classe TextField. Utilisez ensuite la propriété `textFlow` pour appliquer le formatage avancé à partir des classes de TLF.

Vous pouvez également créer l'occurrence de TLFTextField sur la scène à l'aide de l'outil texte de Flash Professional. Vous pouvez alors utiliser ActionScript pour contrôler le formatage et la mise en forme du champ de texte à l'aide de classes TLF. Pour plus d'informations, voir TLFTextField dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Dans Flex, utilisez les classes TLF. Pour plus d'informations, voir « Utilisation de Text Layout Framework » à la page 441.

Utilisation de Text Layout Framework

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Si vous travaillez dans Flex ou créez des composants texte personnalisés, faites appel aux classes TLF. TLF est une bibliothèque ActionScript 3.0 entièrement intégrée à la bibliothèque textLayout.swc. La bibliothèque TLF contient environ 100 classes et interfaces ActionScript 3.0 réparties dans dix packages. Ces packages sont des sous-packages du package flashx.textLayout.

Classes Text Layout Framework

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les classes TLF sont regroupées en trois catégories :

- Classes de formatage et structures de données
- Classes de rendu
- Classes d'interaction utilisateur

Classes de formatage et structures de données

Les packages suivants contiennent les classes de formatage et structures de données de TLF :

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

La structure de données principale de TLF correspond à la hiérarchie d'enchaînements, définie dans le package d'éléments. Au sein de cette structure, vous pouvez attribuer des styles et attributs à des segments de texte en utilisant le package de formats. Le package de conversion permet de contrôler l'importation et l'exportation du texte dans la structure de données.

Classes de rendu

Les packages suivants contiennent les classes de rendu de TLF :

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

Les classes de ces packages facilitent le rendu du texte affiché par Flash Player. Le package d'usine constitue un moyen simple d'afficher du texte statique. Le package de conteneur inclut les classes et les interfaces qui définissent les conteneurs d'affichage du texte dynamique. Le package de composition définit les techniques de positionnement et d'affichage de texte dynamique dans les conteneurs.

Classes d'interaction utilisateur

Les packages suivants contiennent les classes d'interaction utilisateur de TLF :

- flashx.textLayout.edit
- flashx.textLayout.operations
- flashx.textLayout.events

Les packages de modification et d'opérations définissent les classes dont vous disposez pour autoriser la modification du texte stocké dans les structures de données. Le package des événements contient les classes de gestion des événements.

Procédure générale de création de texte à l'aide de Text Layout Framework

La procédure suivante décrit le processus général de création de texte à l'aide de Text Layout Format :

1 Importez du texte formaté dans les structures de données TLF. Pour plus d'informations, voir « Structuration du texte à l'aide de TLF » à la page 446 et « Formatage du texte à l'aide de TLF » à la page 450.

2 Créez un ou plusieurs conteneurs d'objets d'affichage liés destinés au texte. Pour plus d'informations, voir « Gestion des conteneurs de texte à l'aide de TLF » à la page 452.

3 Associez le texte figurant dans les structures de données aux conteneurs et définissez les options de modification et de défilement. Pour plus d'informations, voir « Activation de la sélection, de la modification et de l'annulation de texte à l'aide de TLF » à la page 453.

4 Créez un gestionnaire d'événement permettant de redistribuer le texte en réponse à des événements de redimensionnement (ou autre). Pour plus d'informations, voir « Gestion des événements à l'aide de TLF » à la page 453.

Exemple Text Layout Framework : article de journal

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'exemple suivant illustre la mise en forme d'une page de journal simple à l'aide de TLF. Cette page comprend un gros titre, un sous-titre et une section sur plusieurs colonnes :

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;
    import flash.geom.Rectangle;

    import flashx.textLayout.compose.StandardFlowComposer;
    import flashx.textLayout.container.ContainerController;
    import flashx.textLayout.container.ScrollPolicy;
    import flashx.textLayout.conversion.TextConverter;
    import flashx.textLayout.elements.TextFlow;
    import flashx.textLayout.formats.TextLayoutFormat;

    public class TLFNewsLayout extends Sprite
    {
        private var hTextFlow:TextFlow;
        private var headContainer:Sprite;
        private var headlineController:ContainerController;
        private var hContainerFormat:TextLayoutFormat;

        private var bTextFlow:TextFlow;
        private var bodyTextContainer:Sprite;
        private var bodyController:ContainerController;
        private var bodyTextContainerFormat:TextLayoutFormat;

        private const headlineMarkup:String = "<flow:TextFlow
xmlns:flow='http://ns.adobe.com/textLayout/2008'><flow:p textAlign='center'><flow:span
fontFamily='Helvetica' fontSize='18'>TLF News Layout Example</flow:span><flow:br/><flow:span
fontFamily='Helvetica' fontSize='14'>This example formats text like a newspaper page with a
headline, a subtitle, and multiple columns</flow:span></flow:p></flow:TextFlow>";

        private const bodyMarkup:String = "<flow:TextFlow
xmlns:flow='http://ns.adobe.com/textLayout/2008' fontSize='12' textIndent='10' marginBottom='15'
paddingTop='4' paddingLeft='4'><flow:p marginBottom='inherit'><flow:span>There are many
</flow:span><flow:span fontStyle='italic'>such</flow:span><flow:span> lime-kilns in that tract
of country, for the purpose of burning the white marble which composes a large part of the
```

```
substance of the hills. Some of them, built years ago, and long deserted, with weeds growing in
the vacant round of the interior, which is open to the sky, and grass and wild-flowers rooting
themselves into the chinks of the stones, look already like relics of antiquity, and may yet be
overspread with the lichens of centuries to come. Others, where the lime-burner still feeds his
daily and nightlong fire, afford points of interest to the wanderer among the hills, who seats
himself on a log of wood or a fragment of marble, to hold a chat with the solitary man. It is a
lonesome, and, when the character is inclined to thought, may be an intensely thoughtful
occupation; as it proved in the case of Ethan Brand, who had mused to such strange purpose, in
days gone by, while the fire in this very kiln was burning.</flow:span></flow:p><flow:p
marginBottom='inherit'><flow:span>The man who now watched the fire was of a different order, and
troubled himself with no thoughts save the very few that were requisite to his business. At
frequent intervals, he flung back the clashing weight of the iron door, and, turning his face
from the insufferable glare, thrust in huge logs of oak, or stirred the immense brands with a
long pole. Within the furnace were seen the curling and riotous flames, and the burning marble,
almost molten with the intensity of heat; while without, the reflection of the fire quivered on
the dark intricacy of the surrounding forest, and showed in the foreground a bright and ruddy
little picture of the hut, the spring beside its door, the athletic and coal-begrimed figure of
the lime-burner, and the half-frightened child, shrinking into the protection of his father's
shadow. And when again the iron door was closed, then reappeared the tender light of the half-
full moon, which vainly strove to trace out the indistinct shapes of the neighboring mountains;
and, in the upper sky, there was a flitting congregation of clouds, still faintly tinged with the
rosy sunset, though thus far down into the valley the sunshine had vanished long and long
ago.</flow:span></flow:p></flow:TextFlow>";

        public function TLFNewsLayout()
        {
            //wait for stage to exist
            addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
        }

        private function onAddedToStage(evtObj:Event):void
        {
            removeEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            // Headline text flow and flow composer
            hTextFlow = TextConverter.importToFlow(headlineMarkup,
TextConverter.TEXT_LAYOUT_FORMAT);

            // initialize the headline container and controller objects
            headContainer = new Sprite();
            headlineController = new ContainerController(headContainer);
            headlineController.verticalScrollPolicy = ScrollPolicy.OFF;
            hContainerFormat = new TextLayoutFormat();
            hContainerFormat.paddingTop = 4;
            hContainerFormat.paddingRight = 4;
            hContainerFormat.paddingBottom = 4;
            hContainerFormat.paddingLeft = 4;

            headlineController.format = hContainerFormat;
            hTextFlow.flowComposer.addController(headlineController);
            addChild(headContainer);
            stage.addEventListener(flash.events.Event.RESIZE, resizeHandler);

            // Body text TextFlow and flow composer
            bTextFlow = TextConverter.importToFlow(bodyMarkup,
```

```
TextConverter.TEXT_LAYOUT_FORMAT);

            // The body text container is below, and has three columns
            bodyTextContainer = new Sprite();
            bodyController = new ContainerController(bodyTextContainer);
            bodyTextContainerFormat = new TextLayoutFormat();
            bodyTextContainerFormat.columnCount = 3;
            bodyTextContainerFormat.columnGap = 30;

            bodyController.format = bodyTextContainerFormat;
            bTextFlow.flowComposer.addController(bodyController);
            addChild(bodyTextContainer);
            resizeHandler(null);
        }

        private function resizeHandler(event:Event):void
        {
            const verticalGap:Number = 25;
            const stagePadding:Number = 16;
            var stageWidth:Number = stage.stageWidth - stagePadding;
            var stageHeight:Number = stage.stageHeight - stagePadding;
            var headlineWidth:Number = stageWidth;
            var headlineContainerHeight:Number = stageHeight;

            // Initial compose to get height of headline after resize
            headlineController.setCompositionSize(headlineWidth,
headlineContainerHeight);
            hTextFlow.flowComposer.compose();
            var rect:Rectangle = headlineController.getContentBounds();
            headlineContainerHeight = rect.height;

            // Resize and place headline text container
            // Call setCompositionSize() again with updated headline height
            headlineController.setCompositionSize(headlineWidth, headlineContainerHeight );
            headlineController.container.x = stagePadding / 2;
            headlineController.container.y = stagePadding / 2;
            hTextFlow.flowComposer.updateAllControllers();

            // Resize and place body text container
            var bodyContainerHeight:Number = (stageHeight - verticalGap -
headlineContainerHeight);
            bodyController.format = bodyTextContainerFormat;
            bodyController.setCompositionSize(stageWidth, bodyContainerHeight );
            bodyController.container.x = (stagePadding/2);
            bodyController.container.y = (stagePadding/2) + headlineContainerHeight +
verticalGap;
            bTextFlow.flowComposer.updateAllControllers();
        }
    }
}
```

La classe TLFNewsLayout utilise deux conteneurs de texte. Le premier affiche un titre et un sous-titre, tandis que le second contient trois colonnes de texte. Pour raison de simplicité, le texte est codé en dur dans l'exemple au format TLF Markup. La variable `headlineMarkup` contient le titre et le sous-titre, tandis que la variable `bodyMarkup` contient le corps du texte. Pour plus d'informations sur TLF Markup, voir « Structuration du texte à l'aide de TLF » à la page 446.

Au terme d'une initialisation, la fonction `onAddedToStage()` importe le titre du titre dans un objet TextFlow, qui correspond à la principale structure de données de TLF :

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

Le code suivant permet ensuite de créer un objet Sprite destiné au conteneur, puis de créer un contrôleur et de l'associer au conteneur :

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

Le contrôleur est initialisé de sorte à définir le formatage, le défilement et autres options. Il contient la géométrie qui définit les limites du conteneur dans lequel est distribué le texte. Un objet TextLayoutFormat contient les options de formatage suivantes :

```
hContainerFormat = new TextLayoutFormat();
```

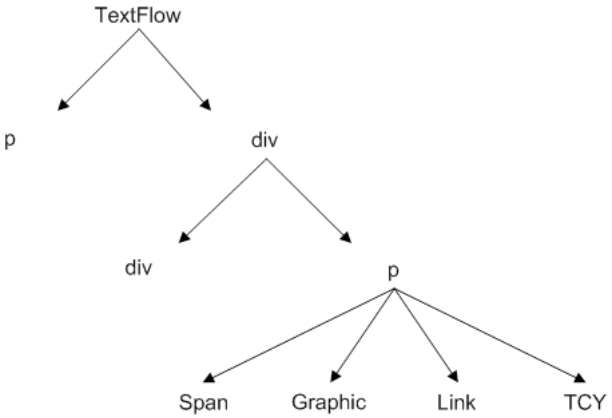
Le contrôleur est affecté au compositeur d'enchaînements et la fonction ajoute le conteneur à la liste d'affichage. La composition et l'affichage à proprement parler des conteneurs relèvent de la méthode `resizeHandler()`. Une procédure identique permet d'initialiser l'objet TextFlow associé au corps du texte.

La méthode `resizeHandler()` mesure l'espace disponible pour le rendu des conteneurs et affecte à ces derniers la taille appropriée. Un appel initial de la méthode `compose()` permet de calculer la hauteur appropriée du conteneur de gros titre. La méthode `resizeHandler()` peut alors placer et afficher le conteneur de gros titre par le biais de la méthode `updateAllControllers()`. Enfin, la méthode `resizeHandler()` se base sur la taille du conteneur de gros titre pour déterminer le positionnement du conteneur de corps de texte.

Structuration du texte à l'aide de TLF

TLF représente le texte sous forme d'arborescence. Chaque nœud de l'arborescence est une occurrence d'une classe définie dans le package d'éléments. Par exemple, le nœud racine de l'arborescence est toujours une occurrence de la classe TextFlow. La class TextFlow représente un article de texte entier. Un article est un ensemble d'éléments texte ou autre assimilés à une seule unité, appelée enchaînement. Un article unique requiert parfois plusieurs colonnes ou conteneurs de texte pour s'afficher.

Exception faite du nœud racine, les éléments restants sont peu ou prou basés sur des éléments XHTML. Le schéma suivant illustre l'arborescence de la structure :



Arborescence TextFlow

Format TLF Markup

Une maîtrise de la structure de TLF s'avère également utile lorsque vous utilisez le format TLF Markup. Le format TLF Markup est une représentation XML du texte stocké dans TLF. Bien que la structure prenne également en charge d'autres formats XML, le format TLF Markup est unique, car il est basé spécifiquement sur la structure de l'arborescence TextFlow. Si vous exportez du code XML à partir d'une arborescence TextFlow à l'aide de ce format de balisage, le code XML est exporté en préservant sa structure arborescente.

TLF Markup assure la représentation du texte la plus fidèle dans une arborescence TextFlow. Le langage de balisage associe des balises à chaque élément de base de l'arborescence TextFlow et fournit également des attributs pour toutes les propriétés de formatage intégrées à la classe TextLayoutFormat.

Le tableau suivant dresse la liste les balises dont vous disposez dans TLF Markup.

Elément	Description	Enfants	Classe
textflow	Elément racine du balisage	div, p	TextFlow
div	Division au sein d'un flux TextFlow. Peut contenir un groupe de paragraphes.	div, list, p	DivElement
p	Paragraphe.	a, tcy, span, img, tab, br, g	ParagraphElement
a	Lien	tcy, span, img, tab, br, g	LinkElement
tcy	Segment de texte horizontal (utilisé dans un élément TextFlow vertical)	a, span, img, tab, br, g	TCYElement
span	Segment de texte au sein d'un paragraphe		SpanElement
img	Image dans un paragraphe		InlineGraphicElement
tab	Caractère de tabulation		TabElement
br	Caractère de saut. Permet d'arrêter une ligne au sein d'un paragraphe. Le texte passe à la ligne suivante, mais sans changer de paragraphe.		BreakElement
linkNormalFormat	Définit les attributs de formatage utilisés pour les liens en état normal.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	Définit les attributs de formatage utilisés pour les liens en état actif, c'est-à-dire lorsque l'utilisateur appuie sur un lien avec le bouton de la souris.	TextLayoutFormat	TextLayoutFormat
linkHoverFormat	Définit les attributs de formatage utilisés pour les liens en état suspendu, c'est-à-dire lorsque l'utilisateur survole un lien avec la souris.	TextLayoutFormat	TextLayoutFormat
li	Composant d'un élément de liste. Doit résider à l'intérieur d'un élément de liste.	div, li, list, p	ListItemElement
list	Liste. Les listes peuvent être imbriquées ou placées les unes à côté des autres. Différents modèles de libellé ou de numérotation peuvent être appliqués aux éléments de liste.	div, li, list, p	ListElement
g	Elément de groupe. Permet de regrouper les éléments dans un paragraphe. Permet d'imbriquer des éléments au-dessous du niveau paragraphe.	a, tcy, span, img, tab, br, g	SubParagraphGroupElement

Voir aussi

TLF 2.0 Lists Markup (disponible en anglais uniquement)

TLF 2.0 SubParagraphGroupElements and typeName (disponible en anglais uniquement)

Utilisation des listes numérotées et des listes à puce

Les classes ListElement et ListItemElement permettent d'ajouter des listes à puce aux contrôles de texte. Il est possible d'imbriquer les listes à puce et de les personnaliser en vue d'utiliser plusieurs puces (ou marqueurs), ainsi que la numérotation automatique et la numérotation de style contour.

Pour créer des listes dans un enchaînement, utilisez la balise `<list>`. Vous utilisez ensuite les balises `<li>` au sein de la balise `<list>` pour chaque élément de la liste. La classe ListMarkerFormat permet de personnaliser l'aspect des puces.

L'exemple suivant crée des listes simples :

```
<flow:list paddingRight="24" paddingLeft="24">
    <flow:li>Item 1</flow:li>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

Comme l'illustre l'exemple ci-dessous, vous pouvez imbriquer des listes au sein d'autres listes :

```
<flow:list paddingRight="24" paddingLeft="24">
    <flow:li>Item 1</flow:li>
    <flow:list paddingRight="24" paddingLeft="24">
        <flow:li>Item 1a</flow:li>
        <flow:li>Item 1b</flow:li>
        <flow:li>Item 1c</flow:li>
    </flow:list>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

Pour personnaliser le type de marques de la liste, utilisez la propriété `listStyleType` de la classe ListElement. Cette propriété peut posséder n'importe quelle valeur définie par la classe ListStyleType (`check`, `circle`, `decimal` et `box`, par exemple). L'exemple suivant permet de créer des listes dotées de divers types de marques et d'un incrément de compteur personnalisé :

```
<flow:list paddingRight="24" paddingLeft="24" listStyleType="upperAlpha">
<flow:li>upperAlpha item</flow:li> <flow:li>another</flow:li> </flow:list> <flow:list
paddingRight="24" paddingLeft="24" listStyleType="lowerAlpha"> <flow:li>lowerAlpha
item</flow:li> <flow:li>another</flow:li> </flow:list> <flow:list paddingRight="24"
paddingLeft="24" listStyleType="upperRoman"> <flow:li>upperRoman item</flow:li>
<flow:li>another</flow:li> </flow:list> <flow:list paddingRight="24" paddingLeft="24"
listStyleType="lowerRoman"> <flow:listMarkerFormat> <!-- Increments the list by 2s rather than
1s. --> <flow:ListMarkerFormat counterIncrement="ordered 2"/> </flow:listMarkerFormat>
<flow:li>lowerRoman item</flow:li> <flow:li>another</flow:li> </flow:list>
```

Vous utilisez la classe ListMarkerFormat pour définir le compteur. Parallèlement à la définition de l'incrément d'un compteur, vous pouvez personnaliser ce dernier en le réinitialisant à l'aide de la propriété `counterReset`.

Vous pouvez personnaliser plus encore l'aspect des marques de liste à l'aide des propriétés `beforeContent` et `afterContent` de la classe ListMarkerFormat. Ces propriétés s'appliquent au contenu affiché avant et après le contenu de la marque.

L'exemple suivant ajoute la chaîne « XX » avant la marque et la chaîne « YY » après ce dernier :

```
<flow:list listStyleType="upperRoman" paddingLeft="36" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="XX"
            afterContent="YY"
            counterIncrement="ordered -1"/>
        </flow:listMarkerFormat>
    <flow:li>Item 1</flow:li>
    <flow:li>Item 2</flow:li>
    <flow:li>Item 3</flow:li>
</flow:list>
```

La propriété `content` définit quant à elle d'autres personnalisations du format de la marque. L'exemple suivant illustre une marque de type chiffre romain en majuscules :

```
<flow:list listStyleType="disc"  paddingLeft="96" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="Section "
            content="counters(ordered,&amp;quot;*&amp;quot;,upperRoman)"
            afterContent=": "/>
    </flow:listMarkerFormat>
    <flow:li>Item 1</li>
    <flow:li>Item 2</li>
    <flow:li>Item 3</li>
</flow:list>
```

Comme illustré par l'exemple précédent, la propriété `content` peut également insérer un suffixe, c'est-à-dire une chaîne insérée après la marque, mais avant la propriété `afterContent`. Pour insérer cette chaîne lorsque vous fournissez un contenu XML à l'enchaînement, entourez-la d'entités HTML `&quote;` plutôt que de guillemets (`"<chaîne>"`).

Voir aussi

TLF 2.0 Lists Markup (disponible en anglais uniquement)

Utilisation de marges dans TLF

Chaque objet FlowElement prend en charge des propriétés de marge destinées à contrôler la position de la zone de contenu de chaque élément, ainsi que l'espace qui sépare les zones de contenu.

La largeur totale d'un élément correspond à la somme de la largeur de son contenu et des propriétés `paddingLeft` et `paddingRight`. La hauteur totale d'un élément correspond à la somme de la hauteur de son contenu et des propriétés `paddingTop` et `paddingBottom`.

La marge correspond à l'espace qui sépare la bordure du contenu. Les propriétés de marge sont `paddingBottom`, `paddingTop`, `paddingLeft` et `paddingRight`. Il est possible d'appliquer un remplissage à l'objet TextFlow, ainsi qu'aux éléments enfants suivants :

- div
- img
- li
- list

- p

Il est impossible d'appliquer les propriétés de marge aux éléments span.

L'exemple suivant définit les propriétés de marge de l'objet TextFlow :

```
<flow:TextFlow version="2.0.0" xmlns:flow="http://ns.adobe.com/textLayout/2008" fontSize="14"
textIndent="15" paddingTop="4" paddingLeft="4" fontFamily="Times New Roman">
```

Les valeurs valides des propriétés de remplissage sont un numéro (en pixels), « auto » et « inherit ». La valeur par défaut est « auto », qui signifie que la marge est automatiquement calculée et définie sur 0 pour tous les éléments, à l'exception de ListElement. Pour les objets ListElement, « auto » est défini sur 0, à l'exception du côté début de la liste, qui utilise la valeur de la propriété `listAutoPadding`. La valeur par défaut de `listAutoPadding` est de 40, qui donne aux listes un retrait par défaut.

Les propriétés de marge n'héritent par défaut d'aucune valeur. Les valeurs « auto » et « inherit » sont des constantes définies par la classe FormatValue.

Les propriétés de marge peuvent être des valeurs négatives.

Voir aussi

Padding changes in TLF 2.0 (disponible en anglais uniquement)

Formatage du texte à l'aide de TLF

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le package flashx.textLayout.formats contient des interfaces et classes qui permettent d'affecter des formats à tout objet FlowElement de l'arborescence d'enchaînements. Il existe deux manières d'appliquer le formatage. Vous pouvez affecter individuellement un format donné ou affecter simultanément un groupe de formats par le biais d'un objet de formatage spécial.

L'interface ITextLayoutFormat rassemble tous les formats susceptibles d'être appliqués à un objet FlowElement. Certains formats s'appliquent à un paragraphe de texte ou conteneur entier mais pas, en toute logique, à des caractères individuels. Ainsi, les formats tels que la justification et les taquets de tabulation s'appliquent à des paragraphes entiers, mais pas à des caractères individuels.

Affectation de formats à une classe FlowElement à l'aide de propriétés

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez affecter des formats à tout objet FlowElement par le biais de propriétés. La classe FlowElement implémente l'interface ITextLayoutFormat, c'est pourquoi toute sous-classe de la classe FlowElement doit également implémenter cette interface.

Ainsi, le code suivant illustre l'affectation de formats individuels à une occurrence de ParagraphElement :

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

Affectation de formats à une classe FlowElement à l'aide de la classe TextLayoutFormat

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Vous pouvez appliquer des formats à un objet FlowElement à l'aide de la classe TextLayoutFormat. Cette classe permet de créer un objet de formatage spécial qui contient toutes les valeurs de formatage requises. Vous pouvez ensuite affecter cet objet à la propriété `format` de n'importe quel objet FlowElement. Les classes TextLayoutFormat et FlowElement implémentent toutes deux l'interface ITextLayoutFormat. Cette caractéristique garantit que les deux classes contiennent les mêmes propriétés de format.

Pour plus d'informations, voir TextLayoutFormat dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Héritage des formats

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les formats sont hérités par le biais de l'arborescence d'enchaînements. Si vous affectez une occurrence de TextLayoutFormat à une occurrence de FlowElement possédant des enfants, la structure déclenche un processus appelé *cascade*. Dans le cadre d'une cascade, la structure examine de manière récursive chaque nœud de la hiérarchie qui hérite de valeurs de l'objet FlowElement. Elle détermine ensuite s'il est nécessaire d'affecter les valeurs héritées à chaque propriété de formatage. La cascade est définie par les règles suivantes :

1. Les valeurs des propriétés sont héritées uniquement d'un ancêtre proche (appelé également le parent).
2. Les valeurs des propriétés sont héritées uniquement si la propriété ne possède aucune valeur (en d'autres termes, la valeur est `non définie`)..
3. Certains attributs non définis n'héritent d'une valeur que si la valeur de l'attribut est définie sur « inherit » ou sur la constante `flashx.textLayout.formats.FormatValue.INHERIT`.

Par exemple, si vous définissez la valeur `fontSize` au niveau du flux TextFlow, ce paramètre s'applique à tous les éléments du flux TextFlow. En d'autres mots, les valeurs sont propagées en cascade vers le bas de l'arborescence d'enchaînements. Vous pouvez cependant remplacer la valeur d'un élément donné en lui affectant directement une nouvelle valeur. A titre de contre-exemple, si vous définissez la valeur `backgroundColor` au niveau TextFlow, les enfants du flux TextFlow n'en héritent pas. La propriété `backgroundColor` n'hérite en effet jamais de ses parents au cours d'une cascade. Vous pouvez éviter ce comportement en définissant la propriété `backgroundColor` de chaque enfant sur la valeur `flashx.textLayout.formats.FormatValue.INHERIT`..

Pour plus d'informations, voir TextLayoutFormat dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Importation et exportation de texte à l'aide de TLF

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La classe TextConverter du package flashx.textLayout.conversion.* permet d'importer du texte dans TLF et d'en exporter hors de ce dernier. Utilisez cette classe si vous envisagez de charger du texte lors de l'exécution au lieu de le compiler dans le fichier SWF. Cette classe sert également à exporter du texte stocké dans une occurrence de TextFlow dans un objet String ou XML.

Les procédures d'importation et d'exportation sont simples. Il vous suffit d'appeler les méthodes `export()` ou `importToFlow()`, qui font toutes deux partie de la classe TextConverter. Ces deux méthodes sont statiques, ce qui signifie que vous les appelez sur la classe TextConverter plutôt que sur une occurrence de cette dernière.

Les classes intégrées au package flashx.textLayout.conversion sont d'une souplesse considérable en ce qui concerne l'emplacement de stockage du texte. Ainsi, si vous stockez le texte dans une base de données, vous pouvez l'importer dans la structure pour l'afficher. Grâce aux classes du package flashx.textLayout.edit, vous pouvez alors modifier le texte et réexporter le texte modifié dans la base de données.

Pour plus d'informations, voir flashx.textLayout.conversion dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Gestion des conteneurs de texte à l'aide de TLF

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Une fois le texte stocké dans les structures de données TLF, Flash Player peut l'afficher. Le texte stocké dans l'arborescence d'enchaînements doit être converti dans un format géré par Flash Player. TLF propose deux méthodes de création d'objets d'affichage à partir d'un enchaînement. La première approche, plus simple, est adaptée à l'affichage de texte statique. La seconde, plus complexe, permet de créer du texte dynamique qui peut être sélectionné et modifié. Dans les deux cas, le texte est finalement converti en occurrences de la classe TextLine, qui fait partie du package flash.text.engine de Flash Player 10.

Création de texte statique

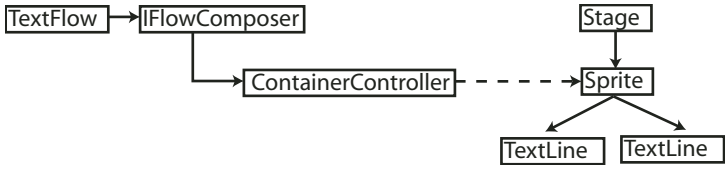
L'approche simple exploite la classe TextFlowTextLineFactory, qui se trouve dans le package flashx.textLayout.factory. L'avantage de cette approche, hormis sa simplicité, est qu'elle limite l'encombrement mémoire par rapport à l'approche FlowComposer. Il est recommandé d'y faire appel pour le texte statique que l'utilisateur ne doit ni modifier, ni sélectionner, ni faire défiler.

Pour plus d'informations, voir TextFlowTextLineFactory dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Création de texte dynamique et de conteneurs

Un compositeur d'enchaînements permet de mieux contrôler l'affichage du texte que la classe TextFlowTextLineFactory. Grâce à un compositeur d'enchaînements, les utilisateurs peuvent, par exemple, sélectionner et modifier le texte. Pour plus d'informations, voir « Activation de la sélection, de la modification et de l'annulation de texte à l'aide de TLF » à la page 453.

Un compositeur d'enchaînements est une occurrence de la classe StandardFlowComposer du package flashx.textLayout.compose. Il gère la conversion d'éléments TextFlow en occurrences de TextLine, ainsi que l'insertion de ces occurrences dans un ou plusieurs conteneurs.



Un compositeur IFlowComposer possède un minimum de zéro objet ContainerController.

Chaque occurrence de TextFlow possède un objet correspondant qui implémente l'interface IFlowComposer. Cet objet IFlowComposer est accessible via la propriété `TextFlow.flowComposer`. Vous pouvez appeler les méthodes définies par l'interface IFlowComposer par le biais de cette propriété. Ces méthodes permettent d'associer le texte à un ou plusieurs conteneurs et de préparer le texte de sorte à l'afficher au sein d'un conteneur.

Un conteneur est une occurrence de la classe Sprite, qui est une sous-classe de la classe DisplayObjectContainer. Ces deux classes font partie intégrante de l'API de liste d'affichage Flash Player. Un conteneur est une forme plus avancée du rectangle de sélection utilisé par la classe TextLineFactory. A l'instar du rectangle de sélection, un conteneur circonscrit la zone dans laquelle s'affichent les occurrences de TextLine. A l'encontre d'un rectangle de sélection, à chaque conteneur correspond un objet de « contrôleur ». Le contrôleur gère le défilement, la composition, la liaison, le formatage et la gestion des événements à l'intention d'un conteneur ou d'un ensemble de conteneurs. A chaque conteneur correspond un objet de contrôleur, qui est une occurrence de la classe ContainerController du package flashx.textLayout.container.

Pour afficher du texte, créez un objet de contrôleur destiné à gérer le conteneur et associez-le au compositeur d'enchaînements. Une fois le conteneur associé, composez le texte pour pouvoir l'afficher. Les conteneurs gèrent donc deux états, composition et affichage. La composition correspond au processus de conversion du texte issu de l'arborescence d'enchaînements en occurrences de TextLine, et de calcul de la position de ces occurrences dans le conteneur. L'affichage correspond au processus de mise à jour de la liste Flash Player.

Pour plus d'informations, voir IFlowComposer, StandardFlowComposer et ContainerController dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Activation de la sélection, de la modification et de l'annulation de texte à l'aide de TLF

Flash Player 9.0 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La possibilité de sélectionner ou modifier du texte est contrôlée au niveau de l'enchaînement. Chaque occurrence de la classe TextFlow est associée à un gestionnaire d'interaction. Vous pouvez accéder au gestionnaire d'interaction d'un objet TextFlow par le biais de la propriété `TextFlow.interactionManager` de ce dernier. Pour activer la sélection de texte, affectez une occurrence de la classe SelectionManager à la propriété `interactionManager`. Pour activer la sélection et la modification du texte, affectez une occurrence de la classe EditManager à la place d'une occurrence de la classe SelectionManager. Pour annuler une opération; créez une occurrence de la classe UndoManager et spécifiez-la en tant qu'argument lorsque vous appelez le constructeur associé à EditManager. La classe UndoManager gère un historique des activités de modification les plus récentes de l'utilisateur et autorise ce dernier à annuler ou rétablir des modifications spécifiques. Ces trois classes font partie du package de modification.

Pour plus d'informations, voir SelectionManager, EditManager et UndoManager dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Gestion des événements à l'aide de TLF

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Les objets TextFlow distribuent des événements dans de nombreuses circonstances, à savoir :

- lors d'un changement de texte ou de mise en forme ;

- avant le début d'une opération ou après la fin de cette dernière ;
- lors d'un changement d'état d'un objet FlowElement ;
- au terme d'une opération de composition.

Pour plus d'informations, voir flashx.textLayout.events dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Voir aussi

TLF FlowElement and LinkElement Events and EventMirrors (disponible en anglais uniquement)

Positionnement des images dans le texte

Pour positionner l'élément InlineGraphicElement dans le texte, vous disposez des propriétés suivantes :

- Propriété `float` de la classe InlineGraphicElement
- Propriété `clearFloats` de la classe FlowElement

La propriété `float` contrôle le positionnement du graphique et du texte qui l'entoure. La propriété `clearFloats` contrôle le positionnement des éléments du paragraphe par rapport à l'élément `flottant`.

Pour contrôler l'emplacement d'une image dans un élément de texte, vous disposez de la propriété `float`. L'exemple suivant insère une image dans un paragraphe et l'aligne à gauche de sorte que le texte l'habille sur la droite :

```
<flow:p paragraphSpaceAfter="15" >Images in a flow are a good thing. For example, here is a
float. It should show on the left: <flow:img float="left" height="50" width="19"
source="../assets/bulldog.jpg"></flow:img> Don't you agree? Another sentence here. Another
sentence here. Another sentence here. Another sentence here. Another sentence here. Another
sentence here. Another sentence here. Another sentence here.</flow:p>
```

Les valeurs valides de la propriété `float` sont « left », « right », « start », « end » et « none ». La classe Float définit ces constantes. La valeur par défaut est « none ».

La propriété `clearFloats` s'avère utile lorsque vous souhaitez ajuster la position de départ des paragraphes suivants qui habilleraient normalement l'image. Supposons, par exemple, que la largeur d'une image excède celle du premier paragraphe. Pour s'assurer que le deuxième paragraphe débute *après* l'image, définissez la propriété `clearFloats`.

L'exemple suivant utilise une image dont la hauteur excède celle du texte dans le premier paragraphe. Pour que le deuxième paragraphe débute après l'image dans le bloc de texte, cet exemple définit la propriété `clearFloats` associée au deuxième paragraphe sur « end ».

```
<flow:p paragraphSpaceAfter="15" >Here is another float, it should show up on the right:
<flow:img float="right" height="50" elementHeight="200" width="19"
source="../assets/bulldog.jpg"></flow:img>We'll add another paragraph that should clear past
it.</flow:p><flow:p clearFloats="end" >This should appear after the previous float on the
right.</flow:p>
```

Les valeurs valides de la propriété `clearFloats` sont « left », « right », « end », « start », « none » et « both ». La classe ClearFloats définit ces constantes. Vous pouvez également définir la propriété `clearFloats` sur « inherit » (constante définie par la classe FormatValue). La valeur par défaut est « none ».

Voir aussi

TLF Floats (disponible en anglais uniquement)

Chapitre 24 : Utilisation du son

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript est conçu pour des applications immersives, interactives, et le son est un élément des applications immersives puissantes souvent ignoré. Vous pouvez ajouter des effets de son à un jeu vidéo, une réaction acoustique à l'interface utilisateur d'une application, ou même créer un programme qui analyse des fichiers mp3 chargés sur Internet, avec du son au coeur de l'application.

Vous pouvez charger des fichiers audio externes et manipuler l'audio incorporé à un fichier SWF. Vous pouvez contrôler l'audio, créer des représentations visuelles des informations de son et capturer le son du microphone d'un utilisateur.

Voir aussi

Package flash.media

flash.events.SampleDataEvent

Principes de base de l'utilisation du son

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les ordinateurs peuvent capturer et coder l'audio numérique (représentation des informations de son de l'ordinateur), le stocker et le récupérer pour le diffuser sur des hauts-parleurs. Il est possible de lire le son à l'aide d'Adobe® Flash® Player ou Adobe® AIR™ et ActionScript.

Lorsque les données audio sont converties au format numérique, elles possèdent différentes caractéristiques (volume du son, son stéréo ou mono). Lorsque vous lisez un son dans ActionScript, vous pouvez régler ces caractéristiques également (augmenter le volume du son ou faire comme s'il provenait d'une certaine direction, par exemple).

Avant de contrôler un son dans ActionScript, les informations de son doivent être chargées dans Flash Player ou AIR. Vous disposez de cinq techniques de chargement des données audio dans Flash Player ou AIR afin de les utiliser avec ActionScript.

- Vous pouvez charger un fichier audio externe tel qu'un fichier mp3 dans le fichier SWF.
- Vous pouvez incorporer directement les informations audio dans le fichier SWF lors de sa création.
- Vous pouvez capturer l'audio à partir d'un microphone connecté à l'ordinateur d'un utilisateur.
- Vous pouvez diffuser l'audio en continu à partir d'un serveur.
- Vous pouvez générer et lire l'audio en mode dynamique.

Lorsque vous chargez des données audio depuis un fichier de son externe, vous pouvez commencer par lire le début du fichier audio pendant le chargement du reste des données audio.

Même s'il existe différents formats de fichier audio utilisés pour coder l'audio numérique, ActionScript 3.0, Flash Player et AIR prennent en charge les fichiers audio stockés au format mp3. Ils ne peuvent pas charger ni lire directement des fichiers audio de formats différents (WAV ou AIFF, par exemple).

Lorsque vous utilisez du son dans ActionScript, vous utilisez probablement plusieurs classes issues du package flash.media. Utilisez la classe Sound pour accéder aux informations audio en chargeant un fichier audio ou en affectant une fonction à un événement pour échantillonner des données de son, puis en démarrant la lecture. Une fois que vous avez démarré la lecture d'un son, Flash Player et AIR vous permettent d'accéder à un objet SoundChannel. Etant donné qu'un fichier audio chargé est un son parmi d'autres que vous lisez sur l'ordinateur d'un utilisateur, chaque son individuel lu utilise son objet SoundChannel ; c'est la sortie combinée de tous les objets SoundChannel mixés qui est lue sur les haut-parleurs de l'ordinateur. Vous utilisez l'occurrence SoundChannel pour contrôler les propriétés du son et arrêter sa lecture. Enfin, si vous souhaitez contrôler l'audio combiné, la classe SoundMixer vous permet de contrôler la sortie mixée.

Vous pouvez également utiliser d'autres classes pour effectuer des tâches plus spécifiques lorsque vous utilisez du son dans ActionScript. Pour plus d'informations sur toutes les classes liées au son, voir « Présentation de l'architecture audio » à la page 456.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants que vous rencontrerez peut-être :

Amplitude Distance d'un point sur la courbe audio à partir de la ligne zéro ou d'équilibre.

Débit Quantité de données codées ou diffusées en continu pour chaque seconde d'un fichier audio. Pour les fichiers mp3, le débit est généralement exprimé en milliers de bits par seconde (kbits/s). Un débit supérieur est souvent synonyme d'une onde acoustique de meilleure qualité.

Mise en mémoire tampon Réception et stockage de données audio avant leur lecture.

MP3 MPEG-1 Audio Layer 3, ou MP3, est un format de compression audio connu.

Balance horizontale Positionnement d'un signal audio entre les canaux gauche et droit dans un champ acoustique stéréo.

Crête Point le plus élevé d'une courbe audio.

Fréquence d'échantillonnage Définit le nombre d'échantillons par seconde extraits d'un signal audio analogique pour créer un signal numérique. La fréquence d'échantillonnage d'un CD audio standard est de 44,1 kHz ou 44 100 échantillons par seconde.

Diffusion en continu Processus consistant à lire les premières sections d'un fichier audio ou vidéo pendant le chargement des dernières sections de ce fichier depuis un serveur.

Volume Intensité d'un son.

Courbe audio Forme d'un graphique des différentes amplitudes d'un signal audio au cours du temps.

Présentation de l'architecture audio

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vos applications peuvent charger des données audio à partir de cinq sources principales :

- Fichiers audio externes chargés lors de l'exécution
- Ressources audio incorporées dans le fichier SWF de l'application
- Données audio issues d'un microphone connecté au système de l'utilisateur
- Données audio diffusées en continu depuis une passerelle multimédia telle que Flash Media Server

- Données audio générées dynamiquement par le biais du gestionnaire d'événement `sampleData`

Vous pouvez charger entièrement les données audio avant leur lecture, ou bien les lire pendant leur chargement.

ActionScript 3.0 prend en charge les fichiers audio stockés au format mp3. Ils ne peuvent pas charger ni lire directement des fichiers audio de formats différents, tels que WAV ou AIFF. Cependant, à partir de Flash Player 9.0.115.0, les fichiers audio AAC peuvent être chargés et lus à l'aide de la classe NetStream. Il s'agit de la même technique que celle utilisée pour le chargement et la lecture de contenu vidéo. Pour plus d'informations sur cette technique, voir « Utilisation de la vidéo » à la page 489.

Vous pouvez utiliser Adobe Flash Professional pour importer des fichiers audio WAV ou AIFF puis les intégrer dans les fichiers SWF de votre application au format mp3. L'outil de programmation Flash vous permet également de compresser des fichiers audio intégrés pour réduire leur taille (mais ceci se fait au détriment de la qualité du son). Pour plus d'informations, voir « Importation de sons » dans *Utilisation de Flash*.

L'architecture audio d'ActionScript 3.0 utilise les classes suivantes dans le package flash.media.

Classe	**Description**
flash.media.Sound	La classe Sound gère le chargement du son, les propriétés de son de base et lance une lecture audio.
flash.media.SoundChannel	Lorsqu'une application lit un objet Sound, un objet SoundChannel est créé pour contrôler la lecture. L'objet SoundChannel contrôle le volume des canaux de lecture gauche et droit du son. Chaque son lu possède son propre objet SoundChannel.
flash.media.SoundLoaderContext	La classe SoundLoaderContext définit le nombre de secondes de mise en mémoire tampon à utiliser lors du chargement d'un son, et indique si Flash Player ou AIR recherche un fichier de régulation sur le serveur lors du chargement d'un fichier. Un objet SoundLoaderContext est utilisé comme paramètre pour la méthode `Sound.load()`.
flash.media.SoundMixer	La classe SoundMixer contrôle les propriétés de lecture et de sécurité de tous les sons dans une application. En effet, plusieurs canaux audio sont mixés au moyen d'un objet SoundMixer commun. Par conséquent, les valeurs de propriété dans l'objet SoundMixer affectent tous les objets SoundChannel en cours de lecture.
flash.media.SoundTransform	La classe SoundTransform contient des valeurs qui contrôlent le volume du son et la balance. Vous pouvez appliquer un objet SoundTransform à un objet SoundChannel individuel, à l'objet SoundMixer global, ou à un objet Microphone, entre autres.
flash.media.ID3Info	Un objet ID3Info contient des propriétés qui représentent les informations de métadonnées ID3 souvent stockées dans des fichiers audio mp3.
flash.media.Microphone	La classe Microphone représente un microphone ou un autre périphérique d'entrée de son connecté à l'ordinateur de l'utilisateur. L'entrée audio issue d'un microphone peut être acheminée vers des haut-parleurs locaux ou envoyée à un serveur distant. L'objet Microphone contrôle le gain, la fréquence d'échantillonnage et d'autres caractéristiques de son propre flux de son.
flash.media.AudioPlaybackMode	La classe AudioPlaybackMode définit les constantes pour la propriété `audioPlaybackMode` de la classe SoundMixer.

Chaque son chargé et lu nécessite sa propre occurrence des classes Sound et SoundChannel. La sortie issue de plusieurs occurrences de SoundChannel est ensuite mixée par la classe SoundMixer globale pendant la lecture.

Les classes Sound, SoundChannel, et SoundMixer ne sont pas utilisées pour les données audio provenant d'un microphone ou d'une transmission de passerelle multimédia en continu (Flash Media Server, par exemple).

Chargement de fichiers audio externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque occurrence de la classe Sound permet de charger et de déclencher la lecture d'une ressource audio spécifique. Une application ne peut pas réutiliser un objet Sound pour charger plusieurs sons. Si elle souhaite charger une nouvelle ressource audio, elle doit créer un objet Sound.

Si vous chargez un fichier audio de petite taille (un son clic à associer à un bouton, par exemple), votre application peut créer un objet Sound qui charge automatiquement le fichier audio, comme indiqué ci-dessous :

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

Le constructeur `Sound()` accepte un objet URLRequest comme premier paramètre. Lorsqu'une valeur pour le paramètre URLRequest est fournie, le nouvel objet Sound commence à charger automatiquement la ressource audio spécifiée.

Dans le meilleur des cas, votre application doit surveiller la progression du chargement du son et rechercher les erreurs pendant le chargement. Par exemple, si le son clic est volumineux, il risque de ne pas être totalement chargé lorsque l'utilisateur clique sur le bouton qui déclenche le son. Si vous tentez de lire un son non chargé, une erreur d'exécution risque de se produire. Il est préférable d'attendre la fin du chargement du son avant de permettre aux utilisateurs d'effectuer des actions risquant de lancer la lecture des sons.

Un objet Sound envoie plusieurs événements différents pendant le chargement du son. Votre application peut écouter ces événements pour suivre la progression du chargement et vérifier que le son est complètement chargé avant la lecture. Le tableau suivant répertorie les événements pouvant être envoyés par un objet Sound.

Evénement	Description
open (`Event.OPEN`)	Envoyé juste avant le début du chargement du son.
progress (`ProgressEvent.PROGRESS`)	Envoyé régulièrement pendant le chargement du son lorsque des données sont reçues du fichier ou du flux.
id3 (`Event.ID3`)	Envoyé lorsque des données ID3 sont disponibles pour un son mp3.
complete (`Event.COMPLETE`)	Envoyé lorsque toutes les données de la ressource audio ont été chargées.
ioError (`IOErrorEvent.IO_ERROR`)	Envoyé lorsqu'un fichier audio est introuvable ou lorsque le chargement est interrompu avant la réception de toutes les données audio.

Le code suivant illustre la lecture d'un son après son chargement:

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var s:Sound = new Sound();
s.addEventListener(Event.COMPLETE, onSoundLoaded);
var req:URLRequest = new URLRequest("bigSound.mp3");
s.load(req);

function onSoundLoaded(event:Event):void
{
    var localSound:Sound = event.target as Sound;
    localSound.play();
}
```

Tout d'abord, l'exemple de code crée un objet Sound sans lui donner de valeur initiale pour le paramètre URLRequest. Ensuite, il écoute l'événement `Event.COMPLETE` issu de l'objet Sound. La méthode `onSoundLoaded()` s'exécute alors lorsque toutes les données audio sont chargées. Puis, il appelle la méthode `Sound.load()` avec une nouvelle valeur URLRequest pour le fichier audio.

La méthode `onSoundLoaded()` s'exécute lorsque le chargement du son est terminé. La propriété `target` de l'objet Event est une référence à l'objet Sound. L'appel à la méthode `play()` de l'objet Sound lance ensuite la lecture du son.

Surveillance du chargement du son

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les fichiers audio peuvent être très volumineux et leur chargement très long. Flash Player et AIR permettent à votre application de lire des sons avant leur chargement complet. Vous pouvez indiquer à l'utilisateur la quantité de données audio ayant été chargées et la quantité de son déjà lue.

La classe Sound envoie deux événements permettant d'afficher facilement la progression du chargement d'un son : `ProgressEvent.PROGRESS` et `Event.COMPLETE`. L'exemple suivant indique comment utiliser ces événements pour afficher les informations de progression concernant le son en cours de chargement :

```
import flash.events.Event;
import flash.events.ProgressEvent;
import flash.media.Sound;
import flash.net.URLRequest;

var s:Sound = new Sound();
s.addEventListener(ProgressEvent.PROGRESS, onLoadProgress);
s.addEventListener(Event.COMPLETE, onLoadComplete);
s.addEventListener(IOErrorEvent.IO_ERROR, onIOError);

var req:URLRequest = new URLRequest("bigSound.mp3");
s.load(req);

function onLoadProgress(event:ProgressEvent):void
{
    var loadedPct:uint = Math.round(100 * (event.bytesLoaded / event.bytesTotal));
    trace("The sound is " + loadedPct + "% loaded.");
}

function onLoadComplete(event:Event):void
{
    var localSound:Sound = event.target as Sound;
    localSound.play();
}
function onIOError(event:IOErrorEvent)
{
    trace("The sound could not be loaded: " + event.text);
}
```

Ce code crée d'abord un objet Sound puis lui ajoute des écouteurs pour les événements `ProgressEvent.PROGRESS` et `Event.COMPLETE`. Une fois que la méthode `Sound.load()` a été appelée et que les premières données sont reçues du fichier audio, un événement `ProgressEvent.PROGRESS` a lieu et déclenche la méthode `onSoundLoadProgress()`.

Le pourcentage des données audio chargées est équivalent à la valeur de la propriété `bytesLoaded` de l'objet ProgressEvent divisé par la valeur de la propriété `bytesTotal`. Les mêmes propriétés `bytesLoaded` et `bytesTotal` sont disponibles sur l'objet Sound également. L'exemple ci-dessus indique les messages relatifs à la progression du chargement du son, mais vous pouvez facilement utiliser les valeurs `bytesLoaded` et `bytesTotal` pour mettre à jour les composants de la barre de progression, tels que ceux intégrés à la structure d'Adobe Flex ou à l'outil de programmation d'Adobe Flash.

Cet exemple indique également comment une application peut reconnaître et répondre à une erreur lors du chargement des fichiers audio. Par exemple, si un fichier audio avec le nom de fichier donné est introuvable, un événement `Event.IO_ERROR` est envoyé par l'objet Sound. Dans le code précédent, la méthode `onIOError()` s'exécute et affiche un message d'erreur court lorsqu'une erreur se produit.

Utilisation des sons intégrés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez des sons intégrés (au lieu de charger du son depuis un fichier externe) surtout dans le cas de fichiers audio de petite taille servant d'indicateurs dans l'interface utilisateur de votre application (des sons qui sont lus lorsque vous cliquez sur des boutons, par exemple).

Lorsque vous incorporez un fichier audio dans votre application, la taille du fichier SWF résultant augmente proportionnellement à la taille du fichier audio. Ceci signifie que lorsque vous incorporez des fichiers volumineux dans votre application, la taille de votre fichier SWF risque de devenir trop importante.

La méthode exacte à utiliser pour incorporer un fichier de police dans le fichier SWF de l'application varie selon l'environnement de développement.

Utilisation d'un fichier audio incorporé dans Flash

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'outil de programmation Flash vous permet d'importer des sons dans un grand nombre de formats audio et de les stocker comme symboles dans la bibliothèque. Vous pouvez ensuite les affecter à des images dans le scénario ou aux images d'un état de bouton, les utiliser avec des comportements ou directement dans du code ActionScript. Cette section décrit comment utiliser des sons incorporés dans du code ActionScript avec l'outil de programmation Flash. Pour plus d'informations sur les autres façons d'utiliser des sons incorporés dans Flash, voir « Importation de sons » dans *Utilisation de Flash*.

Pour intégrer un fichier son à l'aide de l'outil de programmation Flash :

1. Sélectionnez Fichier > Importer > Importer dans la bibliothèque, puis sélectionnez un fichier audio et importez-le.
2. Cliquez avec le bouton droit de la souris sur le nom du fichier importé dans le panneau Bibliothèque, et sélectionnez Propriétés. Activez la case à cocher Exporter pour ActionScript.
3. Dans le champ Classe, entrez un nom à utiliser lorsque vous faites référence à ce son incorporé dans ActionScript. Par défaut, il utilisera le nom du fichier audio dans ce champ. Si le nom du fichier contient un point (comme dans DrumSound.mp3), vous devez le remplacer par DrumSound ; ActionScript n'autorise pas le caractère point dans les noms de classe. Le champ Classe de base devrait encore afficher flash.media.Sound.

4 Cliquez sur OK. Il se peut qu'une boîte de dialogue indiquant qu'une définition pour cette classe est introuvable dans le chemin de classe apparaisse. Cliquez sur OK et continuez. Si vous avez saisi un nom qui ne correspond pas à celui d'une classe contenue dans le chemin de classe de votre application, une nouvelle classe qui hérite de la classe flash.media.Sound est générée automatiquement.

5 Pour utiliser le son incorporé, vous référencez le nom de classe pour ce son dans ActionScript. Par exemple, le code suivant commence par créer une occurrence de la classe DrumSound générée automatiquement :

```
var drum:DrumSound = new DrumSound();
var channel:SoundChannel = drum.play();
```

DrumSound est une sous-classe de la classe flash.media.Sound. Par conséquent, elle hérite des méthodes et des propriétés de la classe Sound, notamment de la méthode `play()` comme indiqué ci-dessus.

Utilisation d'un fichier audio intégré dans Flex

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il existe plusieurs moyens d'intégrer des ressources son dans une application Flex. Vous pouvez par exemple :

- utiliser la balise de métadonnées `[Embed]` dans un script ;
- utiliser la directive `@Embed` dans MXML pour affecter une ressource intégrée en tant que propriété d'un composant, telle que Button ou SoundEffect ;
- utiliser la directive `@Embed` dans un fichier CSS.

Cette section décrit la première option : comment intégrer des sons dans le code ActionScript au sein d'une application Flex à l'aide de la balise de métadonnées `[Embed]`.

Pour intégrer un élément dans le code ActionScript, utilisez la balise de métadonnées `[Embed]`.

Placez le fichier audio dans le dossier source principal ou dans un autre dossier qui se trouve dans le chemin de création de votre projet. Lorsque le compilateur détecte une balise de métadonnées Embed, il crée la classe d'actifs intégrés à votre intention. Vous pouvez accéder à la classe par le biais d'une variable de données de type Class que vous déclarez immédiatement après la balise de métadonnées `[Embed]`.

Le code suivant intègre un son appelé smallSound.mp3 et utilise une variable appelée `soundClass` pour stocker une référence à la classe de ressources intégrées associée à ce son. Le code crée ensuite une occurrence de la classe de ressources intégrées, l'attribue comme une occurrence de la classe Sound et appelle la méthode `play()` sur cette occurrence :

```
package
{
    import flash.display.Sprite;
    import flash.media.Sound;
    import flash.media.SoundChannel;

    public class EmbeddedSoundExample extends Sprite
    {
        [Embed(source="smallSound.mp3")]
        public var soundClass:Class;

        public function EmbeddedSoundExample()
        {
            var smallSound:Sound = new soundClass() as Sound;
            smallSound.play();
        }
    }
}
```

Pour utiliser le son incorporé en vue de définir une propriété d'un composant Flex, elle doit être attribuée comme une occurrence de la classe mx.core.SoundAsset plutôt que comme une occurrence de la classe Sound. Pour obtenir un exemple similaire qui utilise la classe SoundAsset, voir « Classes des éléments incorporés » dans le manuel Formation à ActionScript 3.0.

Utilisation de fichiers audio de lecture en continu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un fichier audio ou un fichier vidéo est lu alors que ses données sont encore en cours de chargement, il est *lu en continu*. Les fichiers audio externes chargés depuis un serveur distant sont souvent lus en continu de façon à ce que l'utilisateur ne doive pas attendre le chargement complet des données audio pour écouter le son.

La propriété `SoundMixer.bufferTime` représente le nombre de millisecondes de données audio que Flash Player ou AIR doit rassembler avant la lecture du son. En d'autres termes, si la propriété `bufferTime` est définie sur 5000, Flash Player ou AIR charge au moins 5 000 millisecondes de données depuis le fichier audio avant le début de la lecture du son. La valeur `SoundMixer.bufferTime` par défaut est 1000.

Votre application peut ignorer la valeur `SoundMixer.bufferTime` globale pour un son individuel en spécifiant explicitement une nouvelle valeur `bufferTime` lors du chargement du son. Pour ignorer la durée du tampon par défaut, créez d'abord une occurrence de la classe SoundLoaderContext, définissez sa propriété `bufferTime`, puis transmettez-la comme paramètre à la méthode `Sound.load()`, comme indiqué ci-dessous :

```
import flash.media.Sound;
import flash.media.SoundLoaderContext;
import flash.net.URLRequest;

var s:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
var context:SoundLoaderContext = new SoundLoaderContext(8000, true);
s.load(req, context);
s.play();
```

Pendant la lecture, Flash Player ou AIR tente de conserver le tampon audio à la même taille ou à une taille supérieure. Si le téléchargement des données audio est plus rapide que la vitesse de la lecture, cette dernière continue sans interruption. Néanmoins, si la vitesse de chargement des données est ralentie en raison des limites du réseau, la tête de lecture peut atteindre la fin du tampon audio. Dans ce cas, la lecture est suspendue mais elle reprend automatiquement lorsque d'autres données audio sont chargées.

Pour savoir si la lecture est suspendue car Flash Player ou AIR attend le chargement des données, utilisez la propriété `Sound.isBuffering`.

Utilisation de données audio générées de façon dynamique

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Remarque : *Flash Player 10 et Adobe AIR 1.5 donnent désormais la possibilité de générer des données audio de façon dynamique.*

Plutôt que de charger ou de diffuser en continu un son existant, vous pouvez générer des données audio de façon dynamique. Vous pouvez générer des données audio lorsque vous affectez un écouteur associé à l'événement `sampleData` d'un objet Sound (l'événement `sampleData` est défini dans la classe SampleDataEvent du package flash.events). Dans cet environnement, l'objet Sound ne charge pas de données audio à partir d'un fichier. Il agit en fait en tant que socket pour les données audio qui lui sont diffusées en continu par l'intermédiaire de la fonction que vous affectez à cet événement.

Lorsque vous ajoutez un écouteur d'événement `sampleData` à un objet Sound, celui-ci demande périodiquement des données à ajouter au tampon audio. Ce tampon contient des données destinées à être lues par l'objet. Une fois appelé, la méthode `play()` de l'objet Sound distribue l'événement `sampleData` lorsqu'il demande de nouvelles données audio. Ceci n'est vrai que si l'objet Sound n'a pas chargé de données mp3 à partir d'un fichier.

L'objet SampleDataEvent comprend une propriété `data`. Dans votre écouteur d'événement, vous écrivez des objets ByteArray dans cet objet `data`. Les tableaux d'octets que vous écrivez dans cet objet s'ajoutent aux données figurant dans le tampon que lit l'objet Sound. Le tableau d'octets que contient le tampon est un flux de valeurs en virgule flottante comprises en -1 et 1. Chaque valeur représente l'amplitude d'un canal unique (gauche ou droit) d'un échantillon audio. Le son est échantillonné à 44 100 échantillons par seconde. Chaque échantillon contient un canal gauche et un canal droit, entrelacés sous forme de données en virgule flottante dans le tableau d'octets.

Dans votre fonction gestionnaire, vous utilisez la méthode `ByteArray.writeFloat()` pour écrire dans la propriété `data` de l'événement `sampleData`. Par exemple, le code suivant génère une onde sinusoïdale :

```
var mySound:Sound = new Sound();
mySound.addEventListener(SampleDataEvent.SAMPLE_DATA, sineWaveGenerator);
mySound.play();
function sineWaveGenerator(event:SampleDataEvent):void
{
    for (var i:int = 0; i < 8192; i++)
    {
        var n:Number = Math.sin((i + event.position) / Math.PI / 4);
        event.data.writeFloat(n);
        event.data.writeFloat(n);
    }
}
```

Lorsque vous appelez `Sound.play()`, l'application commence à appeler votre gestionnaire d'événement pour demander des données audio d'échantillonnage. Elle continue à envoyer des événements pendant la lecture du son jusqu'à ce que vous cessiez de fournir des données ou que vous appeliez la méthode `SoundChannel.stop()`.

La période d'attente de l'événement varie selon les plates-formes et peut encore changer dans les futures versions de Flash Player et AIR. Plutôt que de vous appuyer sur une période d'attente spécifique, calculez-la. Pour calculer la période d'attente, utilisez la formule suivante :

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

Fournissez entre 2 048 et 8 192 échantillons à la propriété `data` de l'objet SampleDataEvent (pour chaque appel du gestionnaire d'événement). Pour des performances optimales, fournissez autant d'échantillons que possible (8 192 au maximum). Moins vous fournissez d'échantillons, plus il est probable que des bruits parasites se feront entendre pendant la lecture. Ce comportement varie selon les plates-formes et peut se produire dans diverses situations, lors du redimensionnement du navigateur, par exemple. Un code qui fonctionne correctement sur une plate-forme lorsque vous fournissez uniquement 2 048 échantillons ne marchera pas aussi bien sur une autre plate-forme. S'il vous faut le plus court délai d'attente possible, envisagez de permettre à l'utilisateur de sélectionner la quantité de données.

Si vous fournissez moins de 2 048 échantillons (par appel de l'écouteur d'événement `sampleData`), l'application s'arrête à l'issue de la lecture des échantillons restants. L'objet SoundChannel distribue alors un événement SoundComplete.

Modification du son issu de données MP3

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La méthode `SoundExtract` vous permet d'extraire des données d'un objet Sound. Vous pouvez utiliser (et modifier) ces données pour accéder en écriture au flux continu dynamique d'un autre objet Sound à des fins de lecture. Ainsi, le code suivant utilise les octets d'un fichier MP3 chargé et les transmet par le biais d'une fonction de filtre, `upOctave()` :

```
var mySound:Sound = new Sound();
var sourceSnd:Sound = new Sound();
var urlReq:URLRequest = new URLRequest("test.mp3");
sourceSnd.load(urlReq);
sourceSnd.addEventListener(Event.COMPLETE, loaded);
function loaded(event:Event):void
{
    mySound.addEventListener(SampleDataEvent.SAMPLE_DATA, processSound);
    mySound.play();
}
function processSound(event:SampleDataEvent):void
{
        var bytes:ByteArray = new ByteArray();
        sourceSnd.extract(bytes, 8192);
        event.data.writeBytes(upOctave(bytes));
}
function upOctave(bytes:ByteArray):ByteArray
{
    var returnBytes:ByteArray = new ByteArray();
    bytes.position = 0;
    while(bytes.bytesAvailable > 0)
    {
        returnBytes.writeFloat(bytes.readFloat());
        returnBytes.writeFloat(bytes.readFloat());
        if (bytes.bytesAvailable > 0)
        {
            bytes.position += 8;
        }
    }
    return returnBytes;
}
```

Limitations relatives aux sons générés

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Lorsque vous utilisez un écouteur d'événement `sampleData` avec un objet Sound, les seules autres méthodes Sound activées sont `Sound.extract()` et `Sound.play()`. L'appel d'autres méthodes ou propriétés donne lieu à une exception. Tous les méthodes et propriétés de l'objet SoundChannel sont toujours activées.

Lecture de sons

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lire un son chargé peut être aussi simple qu'appeler la méthode `Sound.play()` pour un objet Sound, comme suit :

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

Lorsque vous lisez des sons à l'aide d'ActionScript 3.0, vous pouvez effectuer les opérations suivantes :

- Lire un son à partir d'une position de début spécifique
- Interrompre un son et reprendre la lecture ultérieurement à partir de la même position

- Savoir exactement lorsque la lecture d'un son est terminée
- Suivre la progression de la lecture d'un son
- Modifier le volume ou la balance pendant la lecture d'un son

Pour effectuer ces opérations pendant la lecture, utilisez les classes SoundChannel, SoundMixer et SoundTransform.

La classe SoundChannel contrôle la lecture d'un seul son. La propriété `SoundChannel.position` peut être considérée comme une tête de lecture qui indique le point actuel dans les données audio en cours de lecture.

Lorsqu'une application appelle la méthode `Sound.play()`, une occurrence de la classe SoundChannel est créée pour contrôler la lecture.

Votre application peut lire un son à partir d'une position de début spécifique en la transmettant, en termes de millisecondes, comme paramètre `startTime` de la méthode `Sound.play()`. Elle peut également spécifier un nombre fixe de répétitions du son en succession rapide en transmettant une valeur numérique dans le paramètre `loops` de la méthode `Sound.play()`.

Lorsque la méthode `Sound.play()` est appelée avec un paramètre `startTime` et un paramètre `loops`, le son est lu de façon répétée à partir du même point de début chaque fois, comme indiqué dans le code suivant :

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

Dans cet exemple, le son est lu à partir d'un point une seconde après le début du son, trois fois de suite.

Pause et reprise d'un son

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si votre application lit des sons longs (chansons ou podcasts, par exemple), vous pouvez permettre aux utilisateurs d'interrompre et de reprendre leur lecture. Il est impossible d'interrompre littéralement un son pendant la lecture dans ActionScript ; vous pouvez uniquement l'arrêter. Néanmoins, un son peut être lu à partir de n'importe quel point. Vous pouvez enregistrer la position du son au moment de l'arrêt puis le relire ultérieurement à partir de cette position.

Par exemple, supposons que votre code charge et lit un fichier audio de la façon suivante :

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

Lors de la lecture du son, la propriété `SoundChannel.position` indique le point dans le fichier audio qui est en cours de lecture. Votre application peut stocker la valeur de position avant d'arrêter la lecture du son, comme suit :

```
var pausePosition:int = channel.position;
channel.stop();
```

Pour reprendre la lecture du son, transmettez la valeur de position stockée précédemment pour relancer le son à partir du même point d'arrêt précédent.

```
channel = snd.play(pausePosition);
```

Surveillance de la lecture

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Votre application a peut-être besoin de savoir lorsque la lecture d'un son s'arrête afin de lancer la lecture d'un autre son ou d'effacer des ressources utilisées pendant la lecture précédente. La classe SoundChannel envoie un événement `Event.SOUND_COMPLETE` à la fin de la lecture du son. Votre application peut écouter cet événement et effectuer l'action appropriée, comme indiqué ci-dessous :

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("smallSound.mp3");
snd.load(req);

var channel:SoundChannel = snd.play();
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

public function onPlaybackComplete(event:Event)
{
    trace("The sound has finished playing.");
}
```

La classe SoundChannel n'envoie pas d'événements progress pendant la lecture. Pour fournir des informations relatives à la progression de la lecture, votre application peut définir son propre mécanisme de synchronisation et suivre la position de la tête de lecture du son.

Pour calculer le pourcentage d'un son lu, vous pouvez diviser la valeur de la propriété `SoundChannel.position` par la longueur des données audio en cours de lecture :

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

Néanmoins, ce code signale uniquement des pourcentages de lecture précis si les données audio ont été totalement chargées avant le début de la lecture. La propriété `Sound.length` indique la taille des données audio actuellement chargées, et non pas la taille éventuelle du fichier audio entier. Pour suivre la progression de la lecture d'un son diffusé en continu qui est toujours en cours de chargement, votre application doit estimer la taille éventuelle du fichier audio entier et utiliser cette valeur dans ses calculs. Vous pouvez estimer la longueur éventuelle des données audio à l'aide des propriétés `bytesLoaded` et `bytesTotal` de l'objet Sound, comme suit :

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

Le code suivant charge un fichier audio plus volumineux et utilise l'événement `Event.ENTER_FRAME` comme mécanisme de synchronisation pour afficher la progression de la lecture. Il fournit régulièrement des informations sur le pourcentage de lecture, qui est calculé de la façon suivante : la valeur de position actuelle divisée par la longueur totale des données audio :

```
import flash.events.Event;
import flash.media.Sound;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new
    URLRequest("http://av.adobe.com/podcast/csbu_dev_podcast_epi_2.mp3");
snd.load(req);

var channel:SoundChannel;
channel = snd.play();
addEventListener(Event.ENTER_FRAME, onEnterFrame);
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

function onEnterFrame(event:Event):void
{
    var estimatedLength:int =
        Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
    var playbackPercent:uint =
        Math.round(100 * (channel.position / estimatedLength));
    trace("Sound playback is " + playbackPercent + "% complete.");
}

function onPlaybackComplete(event:Event)
{
    trace("The sound has finished playing.");
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Une fois que le chargement des données audio commence, ce code appelle la méthode `snd.play()` et stocke l'objet SoundChannel résultant dans la variable `channel`. Il ajoute ensuite un écouteur d'événement à l'application principale pour l'événement `Event.ENTER_FRAME` et un autre écouteur d'événement à l'objet SoundChannel pour l'événement `Event.SOUND_COMPLETE` qui a lieu à la fin de la lecture.

Chaque fois que l'application atteint une nouvelle image dans son animation, la méthode `onEnterFrame()` est appelée. La méthode `onEnterFrame()` estime la longueur totale du fichier audio en fonction de la quantité de données déjà chargées puis calcule et affiche le pourcentage de lecture actuel.

Une fois que tout le son a été lu, la méthode `onPlaybackComplete()` s'exécute, supprimant l'écouteur d'événement pour l'événement `Event.ENTER_FRAME` de façon à ce qu'il ne tente pas d'afficher les mises à jour de progression après la lecture.

L'événement `Event.ENTER_FRAME` peut être envoyé plusieurs fois par seconde. Dans certains cas, vous pouvez ne pas afficher la progression de la lecture aussi fréquemment. Votre application peut alors définir son propre mécanisme de synchronisation à l'aide de la classe flash.util.Timer ; voir « Utilisation des dates et des heures » à la page 1.

Arrêt de sons diffusés en continu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il y a quelque chose d'étrange dans le processus de lecture des sons diffusés en continu, c'est-à-dire ceux qui sont lus pendant leur chargement. Lorsque votre application appelle la méthode `SoundChannel.stop()` sur une occurrence de SoundChannel qui lit un son diffusé en continu, la lecture du son s'arrête pendant une image puis elle relance au début du son sur l'image suivante. Ceci a lieu car le chargement du son est toujours en cours. Pour arrêter à la fois le chargement et la lecture d'un son diffusé en continu, appelez la méthode `Sound.close()`.

Sécurité lors du chargement et de la lecture des sons

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'accès aux données audio par votre application peut être limité selon la fonction de sécurité de Flash Player ou AIR. Chaque son est soumis aux restrictions de deux sandbox de sécurité différents, le sandbox pour le contexte lui-même (le sandbox de contexte), et le sandbox pour l'application ou l'objet qui charge et lit le son (le sandbox propriétaire). Pour le contenu de l'application AIR dans le sandbox de sécurité de l'application, tous les sons, y compris ceux chargés à partir d'autres domaines, sont accessibles au contenu dans le sandbox de sécurité de l'application. Toutefois, le contenu dans d'autres sandbox de sécurité observe les même règles que le contenu qui s'exécute dans Flash Player. Pour plus d'informations sur le modèle de sécurité de Flash Player en général et la définition des sandbox, voir « Sécurité » à la page 1085.

Le sandbox de contexte contrôle si des données audio détaillées peuvent être extraites du son à l'aide de la propriété `id3` ou de la méthode `SoundMixer.computeSpectrum()`. Il ne limite pas le chargement ou la lecture du fichier audio lui-même.

Le domaine d'origine du fichier audio définit les limites de sécurité du sandbox de contexte. Généralement, si un fichier audio se trouve dans le même domaine ou dossier que le fichier SWF de l'application ou de l'objet qui le charge, ce dernier dispose d'un accès total à ce fichier audio. Si le son provient d'un domaine différent par rapport à l'application, il peut être intégré dans le sandbox de contexte à l'aide d'un fichier de régulation.

Votre application peut transmettre un objet SoundLoaderContext avec une propriété `checkPolicyFile` comme paramètre à la méthode `Sound.load()`. Lorsque vous définissez la propriété `checkPolicyFile` sur `true`, vous indiquez à Flash Player ou à AIR de rechercher un fichier de régulation sur le serveur à partir duquel le son est chargé. Si un fichier de régulation existe et qu'il autorise l'accès au domaine du fichier SWF de chargement, ce dernier peut charger le fichier audio, accéder à la propriété `id3` de l'objet Sound et appeler la méthode `SoundMixer.computeSpectrum()` pour les sons chargés.

Le sandbox propriétaire contrôle la lecture locale des sons. L'application ou l'objet qui lance la lecture d'un son définit le sandbox propriétaire.

La méthode `SoundMixer.stopAll()` interrompt les sons dans tous les objets SoundChannel en cours de lecture, tant qu'ils répondent aux critères suivants :

- Les sons ont été démarrés par des objets se trouvant dans le même sandbox propriétaire.
- Les sons sont issus d'une source possédant un fichier de régulation qui autorise l'accès au domaine de l'application ou de l'objet qui appelle la méthode `SoundMixer.stopAll()`.

Cependant, dans une application AIR, le contenu du sandbox de sécurité de l'application (contenu installé avec l'application AIR) n'est pas restreint par ces limites de sécurité.

Pour savoir si la méthode `SoundMixer.stopAll()` interrompra tous les sons lus, votre application peut appeler la méthode `SoundMixer.areSoundsInaccessible()`. Si cette méthode renvoie une valeur `true`, certains des sons lus ne sont pas sous le contrôle du sandbox propriétaire actuel et ne seront pas arrêtés par la méthode `SoundMixer.stopAll()`.

La méthode `SoundMixer.stopAll()` empêche également la tête de lecture de continuer pour tous les sons chargés à partir de fichiers externes. Néanmoins, les sons qui sont incorporés dans des fichiers FLA et associés à des images dans le scénario à l'aide de l'outil de programmation Flash risquent d'être relus si l'animation s'est déplacée sur une nouvelle image.

Contrôle du volume du son et de la balance

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet SoundChannel individuel contrôle les canaux stéréo gauche et droit pour un son. Si un son mp3 est un son mono, les canaux stéréo gauche et droit de l'objet SoundChannel contiennent des courbes audio identiques.

Vous pouvez connaître l'amplitude de chaque canal stéréo du son lu à l'aide des propriétés `leftPeak` et `rightPeak` de l'objet SoundChannel. Ces propriétés indiquent l'amplitude de crête de la courbe audio du son. Elles ne représentent pas le volume de lecture réel. Le volume de lecture réel est une fonction de l'amplitude de l'onde acoustique et des valeurs de volume définies dans l'objet SoundChannel et la classe SoundMixer.

Vous pouvez utiliser la propriété pan d'un objet SoundChannel pour indiquer un niveau de volume différent pour chacun des canaux gauche et droit pendant la lecture. La propriété pan peut avoir une valeur comprise entre -1 et 1, où -1 signifie que le canal gauche lit à volume maximal alors que le canal droit est muet, et 1 signifie que le canal droit lit à volume maximal alors que le canal gauche est muet. Les valeurs numériques comprises entre -1 et 1 définissent des valeurs proportionnelles pour les valeurs des canaux gauche et droit, et une valeur de 0 signifie que les deux canaux lisent à un niveau de volume moyen, équilibré.

L'exemple de code suivant crée un objet SoundTransform avec une valeur de volume de 0,6 et une valeur de balance horizontale de -1 (volume de canal gauche maximal et aucun volume de canal droit). Il transmet l'objet SoundTransform comme paramètre à la méthode `play()`, qui l'applique au nouvel objet SoundTransform créé pour contrôler la lecture.

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var trans:SoundTransform = new SoundTransform(0.6, -1);
var channel:SoundChannel = snd.play(0, 1, trans);
```

Vous pouvez modifier le volume et la balance pendant la lecture d'un son en définissant les propriétés `pan` ou `volume` d'un objet SoundTransform puis en appliquant cet objet comme propriété `soundTransform` d'un objet SoundChannel.

Vous pouvez également définir des valeurs de balance et de volume global pour tous les sons à la fois à l'aide de la propriété `soundTransform` de la classe SoundMixer, comme l'indique l'exemple suivant :

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

Vous pouvez également utiliser un objet SoundTransform pour définir des valeurs de balance et de volume global pour un objet Microphone (voir « Capture de l'entrée de son » à la page 476), et pour des objets Sprite et SimpleButton.

L'exemple suivant modifie la balance horizontale du son du canal gauche au canal droit et de nouveau lors de la lecture du son.

```
import flash.events.Event;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundMixer;
import flash.net.URLRequest;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
snd.load(req);

var panCounter:Number = 0;

var trans:SoundTransform;
trans = new SoundTransform(1, 0);
var channel:SoundChannel = snd.play(0, 1, trans);
channel.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

addEventListener(Event.ENTER_FRAME, onEnterFrame);

function onEnterFrame(event:Event):void
{
    trans.pan = Math.sin(panCounter);
    channel.soundTransform = trans; // or SoundMixer.soundTransform = trans;
    panCounter += 0.05;
}

function onPlaybackComplete(event:Event):void
{
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Ce code commence par charger un fichier audio puis crée un objet SoundTransform avec un volume défini sur 1 (volume maximal) et une balance définie sur 0 (balance équilibrée entre gauche et droite). Il appelle ensuite la méthode `snd.play()` en transmettant l'objet SoundTransform comme paramètre.

Lors de la lecture du son, la méthode `onEnterFrame()` s'exécute de façon répétée. La méthode `onEnterFrame()` utilise la fonction `Math.sin()` pour générer une valeur comprise entre -1 et 1 (plage qui correspond aux valeurs acceptables de la propriété `SoundTransform.pan`). La propriété `pan` de l'objet SoundTransform est définie sur la nouvelle valeur, puis la propriété `soundTransform` du canal est définie pour utiliser l'objet SoundTransform modifié.

Pour exécuter cet exemple, remplacez le nom de fichier bigSound.mp3 par le nom d'un fichier mp3 local. Exécutez ensuite l'exemple. Le volume du canal gauche devrait augmenter quand celui du canal droit diminue, et vice-versa.

Dans cet exemple, le même effet peut être obtenu en définissant la propriété `soundTransform` de la classe SoundMixer. Néanmoins, la balance de tous les sons en cours de lecture est affectée (pas seulement le son lu par cet objet SoundChannel).

Utilisation des métadonnées audio

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les fichiers audio qui utilisent le format mp3 peuvent contenir des données supplémentaires relatives au son sous la forme de balises ID3.

Tous les fichiers mp3 ne contiennent pas de métadonnées ID3. Lorsqu'un objet Sound charge un fichier audio mp3, il envoie un événement `Event.ID3` si le fichier audio contient des métadonnées ID3. Pour éviter des erreurs d'exécution, votre application doit attendre de recevoir l'événement `Event.ID3` avant d'accéder à la propriété `Sound.id3` pour un son chargé.

Le code suivant indique comment savoir si les métadonnées ID3 pour un fichier audio ont été chargées :

```
import flash.events.Event;
import flash.media.ID3Info;
import flash.media.Sound;

var s:Sound = new Sound();
s.addEventListener(Event.ID3, onID3InfoReceived);
s.load("mySound.mp3");

function onID3InfoReceived(event:Event)
{
    var id3:ID3Info = event.target.id3;

    trace("Received ID3 Info:");
    for (var propName:String in id3)
    {
        trace(propName + " = " + id3[propName]);
    }
}
```

Ce code commence par créer un objet Sound et par lui demander d'écouter l'événement `Event.ID3`. Lorsque les métadonnées ID3 du fichier audio sont chargées, la méthode `onID3InfoReceived()` est appelée. La cible de l'objet Event qui est transmise à la méthode `onID3InfoReceived()` est l'objet Sound d'origine. Par conséquent, la méthode obtient ensuite la propriété `id3` de l'objet Sound puis effectue une itération sur toutes ses propriétés appelées pour suivre leurs valeurs.

Accès aux données audio brutes

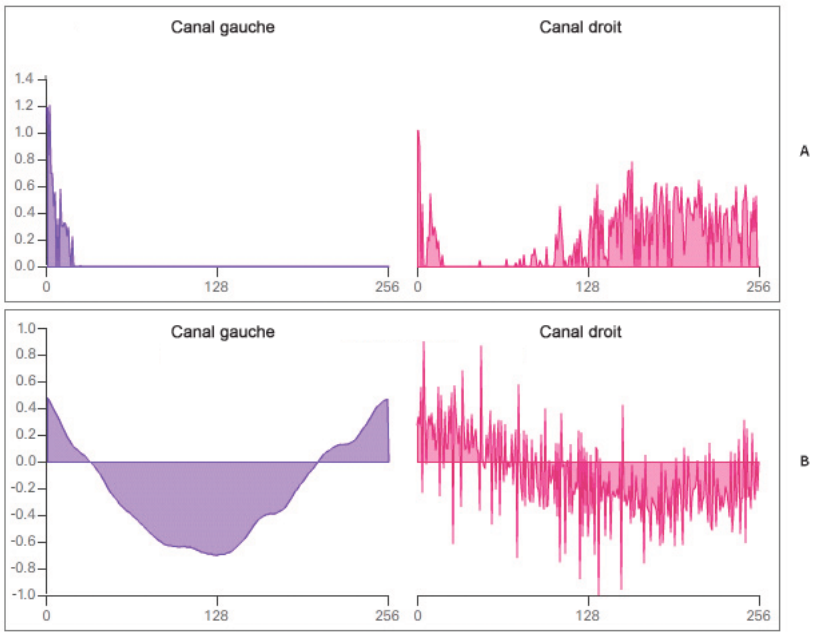
Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `SoundMixer.computeSpectrum()` permet à une application de lire les données audio brutes pour la courbe audio en cours de lecture. Si plusieurs objets SoundChannel sont en cours de lecture, la méthode `SoundMixer.computeSpectrum()` indique les données audio combinées de chaque objet SoundChannel mixé.

Les données audio sont renvoyées sous la forme d'un objet ByteArray contenant 512 octets de données (chacun d'eux contenant une valeur en virgule flottante comprise entre -1 et 1). Ces valeurs représentent l'amplitude des points dans la courbe audio en cours de lecture. Les valeurs sont fournies en deux groupes de 256 : le premier groupe pour le canal stéréo gauche et le second pour le canal stéréo droit.

La méthode `SoundMixer.computeSpectrum()` renvoie des données de spectre de fréquences plutôt que des données de courbe audio si le paramètre `FFTMode` est défini sur `true`. Le spectre de fréquences indique l'amplitude par fréquence du son, de la plus basse à la plus élevée. Une FFT (Fast Fourier Transform - transformation de Fourier rapide) est utilisée pour convertir les données de courbe audio en données de spectre de fréquences. Les valeurs de spectre de fréquences résultantes sont comprises entre 0 et 1,414 environ (la racine carrée de 2).

Le diagramme suivant compare les données renvoyées de la méthode `computeSpectrum()` lorsque le paramètre `FFTMode` est défini sur `true` et lorsqu'il est défini sur `false`. Le son dont les données ont été utilisées pour ce diagramme contient un son de basse de grande intensité dans le canal gauche et un son de tambour dans le canal droit.



Valeurs renvoyées par la méthode SoundMixer.computeSpectrum()
***A.** fftMode=true **B.** fftMode=false*

La méthode `computeSpectrum()` peut également renvoyer des données qui ont été rééchantillonnées à un débit inférieur. Généralement, ceci entraîne des données de fréquence ou des données de courbe audio plus lisses, au profit des détails. Le paramètre `stretchFactor` contrôle la fréquence à laquelle les données de la méthode `computeSpectrum()` sont échantillonnées. Lorsque le paramètre `stretchFactor` est défini sur 0 (valeur par défaut), les données audio sont échantillonnées à une fréquence de 44,1 KHz. La fréquence est diminuée de moitié à chaque valeur successive du paramètre stretchFactor. Par conséquent, une valeur de 1 indique une fréquence de 22,05 KHz, une valeur de 2 une fréquence de 11,025 KHz, et ainsi de suite. La méthode `computeSpectrum()` continue à renvoyer 256 octets par canal stéréo lorsqu'une valeur `stretchFactor` supérieure est utilisée.

La méthode `SoundMixer.computeSpectrum()` comporte des limites :

- Etant donné que les données audio issues d'un microphone ou de flux RTMP ne passent pas par l'objet SoundMixer global, la méthode `SoundMixer.computeSpectrum()` ne renvoie pas de données de ces sources.
- Si un ou plusieurs sons lus proviennent de sources externes au sandbox de contexte actuel, les restrictions de sécurité provoquent le renvoi d'une erreur par la méthode `SoundMixer.computeSpectrum()`. Pour plus d'informations sur les limites de sécurité de la méthode `SoundMixer.computeSpectrum()`, voir « Sécurité lors du chargement et de la lecture des sons » à la page 469 et « Accès aux médias chargés comme s'il s'agissait de données » à la page 1111.

Cependant, dans une application AIR, le contenu du sandbox de sécurité de l'application (contenu installé avec l'application AIR) n'est pas restreint par ces limites de sécurité.

Création d'un visualiseur audio simple

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

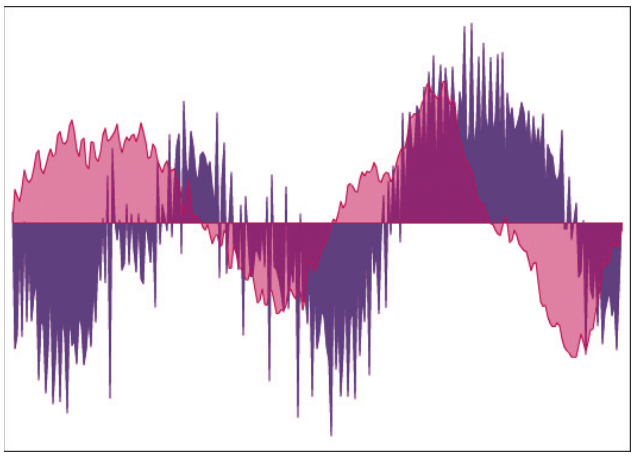
L'exemple suivant utilise la méthode `SoundMixer.computeSpectrum()` pour afficher un diagramme de la courbe audio animée avec chaque image :

```
import flash.display.Graphics;
import flash.events.Event;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundMixer;
import flash.net.URLRequest;

const PLOT_HEIGHT:int = 200;
const CHANNEL_LENGTH:int = 256;

var snd:Sound = new Sound();
var req:URLRequest = new URLRequest("bigSound.mp3");
snd.load(req);

var channel:SoundChannel;
channel = snd.play();
addEventListener(Event.ENTER_FRAME, onEnterFrame);
snd.addEventListener(Event.SOUND_COMPLETE, onPlaybackComplete);

var bytes:ByteArray = new ByteArray();

function onEnterFrame(event:Event):void
{
    SoundMixer.computeSpectrum(bytes, false, 0);

    var g:Graphics = this.graphics;

    g.clear();
    g.lineStyle(0, 0x6600CC);
    g.beginFill(0x6600CC);
    g.moveTo(0, PLOT_HEIGHT);

    var n:Number = 0;

    // left channel
    for (var i:int = 0; i < CHANNEL_LENGTH; i++)
    {
        n = (bytes.readFloat() * PLOT_HEIGHT);
        g.lineTo(i * 2, PLOT_HEIGHT - n);
    }
    g.lineTo(CHANNEL_LENGTH * 2, PLOT_HEIGHT);
```

```
    g.endFill();

    // right channel
    g.lineStyle(0, 0xCC0066);
    g.beginFill(0xCC0066, 0.5);
    g.moveTo(CHANNEL_LENGTH * 2, PLOT_HEIGHT);

    for (i = CHANNEL_LENGTH; i > 0; i--)
    {
        n = (bytes.readFloat() * PLOT_HEIGHT);
        g.lineTo(i * 2, PLOT_HEIGHT - n);
    }
    g.lineTo(0, PLOT_HEIGHT);
    g.endFill();
}

function onPlaybackComplete(event:Event)
{
    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Cet exemple commence par charger et lire un fichier audio puis écoute l'événement `Event.ENTER_FRAME` qui déclenchera la méthode `onEnterFrame()` lors de la lecture du son. La méthode `onEnterFrame()` commence par appeler la méthode `SoundMixer.computeSpectrum()`, qui stocke les données d'onde acoustique dans l'objet ByteArray `bytes`.

La courbe audio est tracée à l'aide de l'API de dessin vectoriel. Une boucle `for` passe dans les 256 premières valeurs de données, représentant le canal stéréo gauche, et trace une ligne entre chaque point au moyen de la méthode `Graphics.lineTo()`. Une second boucle `for` passe dans les 256 valeurs suivantes, en les traçant cette fois dans l'ordre inverse, de droite à gauche. Les tracés de courbe audio résultants peuvent produire un effet miroir-image intéressant, comme illustré sur l'image suivante.



Capture de l'entrée de son

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Microphone permet à votre application de se connecter à un microphone ou à un autre périphérique d'entrée de son sur le système de l'utilisateur et de diffuser l'audio sur les haut-parleurs de ce système ou d'envoyer les données audio à un serveur distant (Flash Media Server, par exemple). Vous pouvez accéder aux données audio brutes à partir du microphone, puis les enregistrer ou les traiter. Vous pouvez également envoyer directement l'audio aux haut-parleurs du système ou envoyer des données audio compressées à un serveur distant. Pour envoyer des données à un serveur distant, vous pouvez utiliser le codec Speex ou Nellymoser (le codec Speex est pris en charge depuis Flash Player version 10 et Adobe AIR version 1.5).

Voir aussi

Michael Chaize : AIR, Android, and the Microphone (disponible en anglais uniquement)

Christophe Coenraets : Voice Notes for Android (disponible en anglais uniquement)

Accès à un microphone

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Microphone ne possède pas de méthode constructeur. A la place, vous utilisez la méthode statique `Microphone.getMicrophone()` pour obtenir une nouvelle occurrence de Microphone, comme indiqué ci-dessous :

```
var mic:Microphone = Microphone.getMicrophone();
```

Lorsque vous appelez la méthode `Microphone.getMicrophone()` sans paramètre, le premier périphérique d'entrée de son détecté sur le système de l'utilisateur est renvoyé.

Un système peut avoir plusieurs périphériques d'entrée de son qui lui sont associés. Votre application peut utiliser la propriété `Microphone.names` pour obtenir un tableau des noms de tous les périphériques d'entrée de son disponibles. Elle peut ensuite appeler la méthode `Microphone.getMicrophone()` avec un paramètre `index` qui correspond à la valeur d'index du nom d'un périphérique dans le tableau.

Il se peut qu'un système n'ait aucun microphone ni périphérique d'entrée de son qui lui soit associé. Vous pouvez utiliser la propriété `Microphone.names` ou la méthode `Microphone.getMicrophone()` pour vérifier si l'utilisateur a installé un périphérique d'entrée de son. Si ce n'est pas le cas, la longueur du tableau `names` est zéro, et la méthode `getMicrophone()` renvoie une valeur `null`.

Lorsque votre application appelle la méthode `Microphone.getMicrophone()`, Flash Player affiche la boîte de dialogue des paramètres de Flash Player, qui invite l'utilisateur à autoriser ou à refuser l'accès Flash Player à la caméra et au microphone sur le système. Une fois que l'utilisateur a fait son choix dans cette boîte de dialogue, un StatusEvent est envoyé. La propriété `code` de cette occurrence de StatusEvent indique si l'accès au microphone a été autorisé ou refusé, comme indiqué dans cet exemple :

```
import flash.media.Microphone;

var mic:Microphone = Microphone.getMicrophone();
mic.addEventListener(StatusEvent.STATUS, this.onMicStatus);

function onMicStatus(event:StatusEvent):void
{
    if (event.code == "Microphone.Unmuted")
    {
        trace("Microphone access was allowed.");
    }
    else if (event.code == "Microphone.Muted")
    {
        trace("Microphone access was denied.");
    }
}
```

La propriété `StatusEvent.code` contiendra Microphone.Unmuted si l'accès a été autorisé, ou Microphone.Muted s'il a été refusé.

la propriété `Microphone.muted` est définie sur `true` ou sur `false` lorsque l'utilisateur autorise ou refuse l'accès au microphone, respectivement. Néanmoins, la propriété `muted` n'est pas définie sur l'occurrence de Microphone tant que StatusEvent n'a pas été distribué. Par conséquent, l'application doit également attendre la distribution de l'événement `StatusEvent.STATUS` avant de vérifier la propriété `Microphone.muted`.

Pour que Flash Player affiche la boîte de dialogue de paramétrage, la taille de la fenêtre de l'application doit être suffisamment élevée (215 sur 138 pixels au moins). Dans le cas contraire, l'accès est automatiquement refusé.

Un contenu qui s'exécute dans le sandbox de l'application AIR ne nécessite pas l'autorisation de l'utilisateur pour accéder au microphone. Par conséquent, les événements d'état associés à l'activation et à la désactivation du microphone ne sont jamais distribués. Etant donné qu'un contenu qui s'exécute dans AIR en dehors du sandbox de l'application ne nécessite pas l'autorisation de l'utilisateur, il est possible de distribuer ces événements d'état.

Acheminement de l'audio du microphone vers des haut-parleurs locaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'entrée audio issue d'un microphone peut être acheminée vers les haut-parleurs du système local en appelant la méthode `Microphone.setLoopback()` avec une valeur de paramètre `true`.

Lorsque le son provenant d'un microphone local est acheminé vers des haut-parleurs locaux, vous risquez de créez une boucle de réaction acoustique pouvant entraîner des grincements d'une grande intensité et endommager ainsi votre matériel. Vous pouvez appeler la méthode `Microphone.setUseEchoSuppression()` avec une valeur de paramètre `true` pour réduire (sans éliminer complètement) le risque de réaction acoustique. Adobe vous conseille de toujours appeler `Microphone.setUseEchoSuppression(true)` avant d'appeler `Microphone.setLoopback(true)`, à moins que vous soyez sûr que l'utilisateur lit le son à l'aide d'un casque ou d'un dispositif autre que les haut-parleurs.

Le code suivant indique comment acheminer l'audio d'un microphone local vers les haut-parleurs du système local :

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

Modification de l'audio du microphone

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Votre application peut modifier les données audio provenant d'un microphone de deux façons différentes. Premièrement, elle peut modifier le gain du son entré, qui multiplie les valeurs d'entrée par une quantité spécifiée pour créer un son plus ou moins intense. La propriété `Microphone.gain` accepte des valeurs numériques comprises entre 0 et 100 inclus. Une valeur de 50 a un rôle de multiplicateur de un et spécifie un volume normal. Une valeur de zéro agit comme un multiplicateur de zéro et interrompt l'audio d'entrée. Les valeurs supérieures à 50 indiquent un volume supérieur à la normale.

Votre application peut également modifier la fréquence d'échantillonnage de l'audio d'entrée. Des fréquences d'échantillonnage supérieures augmentent la qualité du son, mais créent également des flux de données plus denses qui utilisent davantage de ressources pour la transmission et le stockage. La propriété `Microphone.rate` représente la fréquence d'échantillonnage audio mesurée en kilohertz (kHz). La fréquence d'échantillonnage par défaut est de 8 kHz. Vous pouvez définir la propriété `Microphone.rate` sur une valeur supérieure à 8 kHz si votre microphone prend en charge la fréquence supérieure. Par exemple, si vous définissez la propriété `Microphone.rate` sur la valeur 11, la fréquence d'échantillonnage est réglée sur 11 kHz ; si vous la définissez sur 22, la fréquence d'échantillonnage est réglée sur 22 kHz, et ainsi de suite. Les fréquences d'échantillonnage disponibles varient en fonction du codec sélectionné. Lorsque vous utilisez le codec Nellymoser, vous pouvez spécifier les fréquences d'échantillonnage 5, 8, 11, 16, 22 et 44 kHz. Lorsque vous utilisez le codec Speex (disponible à partir de Flash Player 10 et Adobe AIR 1.5), vous ne pouvez utiliser que 16 kHz.

Détection de l'activité du microphone

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour économiser les ressources de traitement et de bande passante, Flash Player tente de détecter lorsque aucun son n'est transmis par un microphone. Lorsque le niveau d'activité du microphone se situe sous le seuil de niveau de silence pendant longtemps, Flash Player arrête la transmission de l'entrée audio et envoie un simple événement ActivityEvent à la place. Si vous utilisez le codec Speex (disponible dans Flash Player 10 et versions ultérieures, ainsi que dans Adobe AIR 1.5 et versions ultérieures), définissez le niveau de silence sur 0 afin de vous assurer que l'application transmet les données audio en continu. La fonction de détection d'activité vocale de Speex réduit automatiquement la bande passante.

Remarque : *un objet Microphone ne distribue des événements Activity que si l'application contrôle actuellement le microphone. Par conséquent, si vous n'appelez pas* `setLoopBack( true )`*, que vous n'associez pas d'écouteur aux événements de données d'exemple ou que vous ne connectez pas le microphone à un objet NetStream, aucun événement d'activité n'est distribué.*

Trois propriétés de la classe Microphone surveillent et contrôlent la détection d'activité :

- La propriété `activityLevel` en lecture seule indique la quantité de son détectée par le microphone, sur une échelle de 0 à 100.
- La propriété `silenceLevel` spécifie la quantité de son nécessaire pour activer le microphone et envoie un événement `ActivityEvent.ACTIVITY`. La propriété `silenceLevel` utilise également une échelle de 0 à 100, et la valeur par défaut est 10.
- La propriété `silenceTimeout` décrit le nombre de millisecondes pendant lequel le niveau d'activité doit rester sous le niveau de silence, jusqu'à ce qu'un événement `ActivityEvent.ACTIVITY` soit envoyé pour indiquer que le microphone est maintenant désactivé. La valeur `silenceTimeout` par défaut est 2000.

Les propriétés `Microphone.silenceLevel` et `Microphone.silenceTimeout` sont en lecture seule, mais vous pouvez modifier leurs valeurs à l'aide de la méthode `Microphone.setSilenceLevel()`.

Dans certains cas, l'activation du microphone alors qu'une nouvelle activité est détectée peut entraîner un court délai. Vous pouvez laisser le microphone actif en permanence pour supprimer ces délais d'activation. Votre application peut appeler la méthode `Microphone.setSilenceLevel()` avec le paramètre `silenceLevel` défini sur zéro pour indiquer à Flash Player de laisser le microphone actif et de continuer à rassembler des données audio, même lorsque aucun son n'est détecté. Inversement, lorsque vous définissez le paramètre `silenceLevel` sur 100, le microphone n'est pas activé.

L'exemple suivant affiche les informations relatives au microphone et aux événements activity et status envoyés par un objet Microphone :

```
import flash.events.ActivityEvent;
import flash.events.StatusEvent;
import flash.media.Microphone;

var deviceArray:Array = Microphone.names;
trace("Available sound input devices:");
for (var i:int = 0; i < deviceArray.length; i++)
{
    trace(" " + deviceArray[i]);
}

var mic:Microphone = Microphone.getMicrophone();
mic.gain = 60;
mic.rate = 11;
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
mic.setSilenceLevel(5, 1000);

mic.addEventListener(ActivityEvent.ACTIVITY, this.onMicActivity);
mic.addEventListener(StatusEvent.STATUS, this.onMicStatus);

var micDetails:String = "Sound input device name: " + mic.name + '\n';
micDetails += "Gain: " + mic.gain + '\n';
micDetails += "Rate: " + mic.rate + " kHz" + '\n';
micDetails += "Muted: " + mic.muted + '\n';
micDetails += "Silence level: " + mic.silenceLevel + '\n';
micDetails += "Silence timeout: " + mic.silenceTimeout + '\n';
micDetails += "Echo suppression: " + mic.useEchoSuppression + '\n';
trace(micDetails);

function onMicActivity(event:ActivityEvent):void
{
    trace("activating=" + event.activating + ", activityLevel=" +
        mic.activityLevel);
}

function onMicStatus(event:StatusEvent):void
{
    trace("status: level=" + event.level + ", code=" + event.code);
}
```

Lorsque vous exécutez l'exemple ci-dessus, parlez ou faites du bruit dans votre microphone système et observez les instructions trace qui apparaissent dans une console ou une fenêtre de débogage.

Envoi d'audio vers et depuis une passerelle multimédia

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

D'autres fonctionnalités audio sont disponibles lorsque vous utilisez ActionScript avec une passerelle multimédia de diffusion en continu telle que Flash Media Server.

Votre application peut notamment associer un objet Microphone à un objet NetStream et transmettre directement des données du microphone de l'utilisateur au serveur. Les données audio peuvent également être diffusées en continu du serveur vers une application et lues dans le cadre d'un MovieClip ou au moyen d'un objet Video.

Le codec Speex est pris en charge depuis Flash Player version 10 et Adobe AIR version 1.5. Pour définir le codec utilisé pour les données audio compressées envoyées au serveur multimédia, définissez la propriété `codec` de l'objet Microphone. Cette propriété gère deux valeurs, qui sont énumérées dans la classe SoundCodec. La définition de la propriété codec sur `SoundCodec.SPEEX` sélectionne le codec Speex pour la compression audio. La définition de la propriété codec sur `SoundCodec.NELLYMOSER` (valeur par défaut) sélectionne le codec Nellymoser pour la compression audio.

Pour plus d'informations, voir la documentation de Flash Media Server disponible en ligne à l'adresse suivante : www.adobe.com/go/learn_fms_docs_fr.

Capture des données audio issues d'un microphone

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Dans Flash Player 10.1 et AIR 2 ou ultérieur, vous pouvez capturer les données issues d'un microphone sous forme de tableau d'octets composé de valeurs à virgule flottante. Chaque valeur représente un échantillon de données audio monophoniques.

Pour obtenir les données issues d'un microphone, définissez un écouteur associé à l'événement `sampleData` de l'objet Microphone. L'objet Microphone distribue les événements `sampleData` à fréquence régulière au fur et à mesure que le tampon du microphone se remplit d'échantillons audio. La propriété `data` de l'objet SampleDataEvent correspond à un tableau d'octets d'échantillons audio. Les échantillons sont représentés par des valeurs à virgule flottante, chacune d'elles correspondant à un échantillon audio monophonique.

Le code suivant capture les données audio issues d'un microphone dans un objet ByteArray appelé `soundBytes` :

```
var mic:Microphone = Microphone.getMicrophone();
mic.setSilenceLevel(0, DELAY_LENGTH);
mic.addEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);
function micSampleDataHandler(event:SampleDataEvent):void {
    while(event.data.bytesAvailable) {
        var sample:Number = event.data.readFloat();
        soundBytes.writeFloat(sample);
    }
}
```

Vous pouvez réutiliser les octets d'échantillon sous forme d'audio à lire d'un objet Sound. Pour ce faire, définissez la propriété `rate` de l'objet Microphone sur 44, à savoir la fréquence d'échantillonnage utilisée par les objets Sound. (Vous pouvez également convertir des échantillons issus de microphone capturés à une fréquence inférieure sur le taux 44 kHz requis par l'objet Sound.) N'oubliez pas non plus que l'objet Microphone capture des échantillons monophoniques, alors que l'objet Sound utilise un son stéréo. Ecrivez donc deux fois dans l'objet Sound chaque octet capturé par l'objet Microphone. L'exemple suivant capture 4 secondes de données issues d'un microphone et les lit par le biais d'un objet Sound :

```
const DELAY_LENGTH:int = 4000;
var mic:Microphone = Microphone.getMicrophone();
mic.setSilenceLevel(0, DELAY_LENGTH);
mic.gain = 100;
mic.rate = 44;
mic.addEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);

var timer:Timer = new Timer(DELAY_LENGTH);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.start();

function micSampleDataHandler(event:SampleDataEvent):void
{
    while(event.data.bytesAvailable)
    {
        var sample:Number = event.data.readFloat();
        soundBytes.writeFloat(sample);
    }
}
var sound:Sound = new Sound();
var channel:SoundChannel;
function timerHandler(event:TimerEvent):void
{
    mic.removeEventListener(SampleDataEvent.SAMPLE_DATA, micSampleDataHandler);
    timer.stop();
    soundBytes.position = 0;
    sound.addEventListener(SampleDataEvent.SAMPLE_DATA, playbackSampleHandler);
    channel.addEventListener( Event.SOUND_COMPLETE, playbackComplete );
    channel = sound.play();
}

function playbackSampleHandler(event:SampleDataEvent):void
{
    for (var i:int = 0; i < 8192 && soundBytes.bytesAvailable > 0; i++)
    {
        trace(sample);
        var sample:Number = soundBytes.readFloat();
        event.data.writeFloat(sample);
        event.data.writeFloat(sample);
    }
}

function playbackComplete( event:Event ):void
{
    trace( "Playback finished.");
}
```

Pour plus d'informations sur la lecture de données d'échantillons audio, voir « Utilisation de données audio générées de façon dynamique » à la page 463.

Exemple d'objet Sound : Podcast Player

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un podcast est un fichier audio distribué sur Internet, sur demande ou sur abonnement. Les podcasts sont généralement publiés dans un annuaire. Etant donné que les épisodes de podcast peuvent durer d'une minute à plusieurs heures, ils sont généralement diffusés en continu pendant la lecture. Les épisodes de podcast, également appelés éléments, sont généralement fournis au format de fichier mp3. Les podcasts vidéo sont également courants, mais cet exemple d'application lit uniquement des podcasts audio utilisant des fichiers mp3.

Cet exemple n'est pas une application agrégatrice de podcasts comprenant toutes les fonctionnalités. Par exemple, elle ne gère pas les abonnements à des podcasts spécifiques et ne mémorise pas les podcasts qu'un utilisateur a écoutés lors de l'exécution suivante de l'application. Il peut servir de point de départ pour un agrégateur de podcasts comprenant toutes les fonctionnalités.

L'exemple Podcast Player illustre les techniques de programmation ActionScript suivantes :

- Lecture d'un fil de syndication et analyse de son contenu XML
- Création d'une classe SoundFacade pour simplifier le chargement et la lecture des fichiers audio
- Affichage de la progression de la lecture du son
- Interruption et reprise de la lecture du son

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application Podcast Player se trouvent dans le dossier Samples/PodcastPlayer. L'application se compose des fichiers suivants :

Fichier	Description
PodcastPlayer.mxml ou PodcastPlayer.fla	Interface utilisateur de l'application pour Flex (MXML) ou Flash (FLA).
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	Classe Document contenant la logique de l'interface utilisateur pour le lecteur de podcast (Flash uniquement).
SoundPlayer.mxml	Un composant MXML qui affiche les commandes de lecture et les barres de progression, et contrôle la lecture du son, pour Flex uniquement.
main.css	Styles associés à l'interface utilisateur de l'application (Flex uniquement)
image/	Icônes permettant le formatage des boutons (Flex uniquement).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	Classe pour le symbole du clip SoundPlayer contenant la logique de l'interface utilisateur du lecteur de sons (Flash uniquement).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	Composant de rendu de cellule personnalisé permettant d'afficher un bouton de lecture dans une cellule de la grille de données (Flash uniquement).
com/example/programmingas3/podcastplayer/RSSBase.as	Une classe de base qui fournit les méthodes et les propriétés courantes pour la classe RSSChannel et la classe RSSItem.

Fichier	Description
com/example/programmingas3/podcastplayer/RSSChannel.as	Une classe ActionScript qui contient des données relatives à un canal RSS.
com/example/programmingas3/podcastplayer/RSSItem.as	Une classe ActionScript qui contient des données relatives à un élément RSS.
com/example/programmingas3/podcastplayer/SoundFacade.as	La classe ActionScript principale pour l'application. Elle encapsule les méthodes et les événements des classes Sound et SoundChannel et ajoute une prise en charge pour l'interruption et la reprise de la lecture.
com/example/programmingas3/podcastplayer/URLService.as	Une classe ActionScript qui récupère des données d'une URL distante.
playerconfig.xml	Un fichier XML contenant une liste des fils de syndication qui représentent des chaînes de podcast.
comp/example/programmingas3/utils/DateUtil.as	Classe permettant le formatage rapide de la date (Flash uniquement).

Lecture de données RSS pour une chaîne de podcast

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application Podcast Player commence par lire les informations concernant des chaînes de podcasts et leurs épisodes :

1. L'application commence par lire un fichier de configuration XML qui contient une liste des chaînes de podcast et affiche la liste des chaînes pour l'utilisateur.

2. Lorsque l'utilisateur sélectionne l'une des chaînes de podcast, il lit le flux RSS pour la chaîne et affiche une liste des épisodes de chaîne.

Cet exemple utilise la classe d'utilitaire URLLoader pour récupérer des données de texte depuis un emplacement distant ou un fichier local. L'application Podcast Player crée d'abord un objet URLLoader pour obtenir une liste des fils de syndication au format XML du fichier playerconfig.xml. Ensuite, lorsque l'utilisateur sélectionne un fil de syndication spécifique dans la liste, un nouvel objet URLLoader est créé pour lire les données RSS de l'URL de ce fil.

Simplification de la lecture et du chargement du son à l'aide de la classe SoundFacade

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'architecture audio ActionScript 3.0 est puissante mais complexe. Les applications nécessitant des fonctions de lecture et de chargement de son de base uniquement peuvent utiliser une classe masquant une partie de la complexité en fournissant un ensemble d'appels et d'événements plus simple. Dans l'univers des modèles de conception de logiciel, une telle classe est appelée *façade*.

La classe SoundFacade présente une seule interface permettant d'effectuer les tâches suivantes :

- Chargement de fichiers audio à l'aide d'un objet Sound, d'un objet SoundLoaderContext et d'une classe SoundMixer
- Lecture de fichiers audio à l'aide des objets Sound et SoundChannel
- Envoi d'événements de progression de la lecture

- Interruption et reprise de la lecture du son à l'aide des objets Sound et SoundChannel

La classe SoundFacade essaie d'offrir le meilleur de la fonctionnalité des classes de son ActionScript avec moins de complexité.

Le code suivant indique la déclaration de classe, les propriétés de classe et la méthode constructeur `SoundFacade()` :

```
public class SoundFacade extends EventDispatcher
{
    public var s:Sound;
    public var sc:SoundChannel;
    public var url:String;
    public var bufferTime:int = 1000;

    public var isLoaded:Boolean = false;
    public var isReadyToPlay:Boolean = false;
    public var isPlaying:Boolean = false;
    public var isStreaming:Boolean = true;
    public var autoLoad:Boolean = true;
    public var autoPlay:Boolean = true;

    public var pausePosition:int = 0;

    public static const PLAY_PROGRESS:String = "playProgress";
    public var progressInterval:int = 1000;
    public var playTimer:Timer;

    public function SoundFacade(soundUrl:String, autoLoad:Boolean = true,
                                autoPlay:Boolean = true, streaming:Boolean = true,
                                bufferTime:int = -1):void
    {
        this.url = soundUrl;

        // Sets Boolean values that determine the behavior of this object
        this.autoLoad = autoLoad;
        this.autoPlay = autoPlay;
        this.isStreaming = streaming;

        // Defaults to the global bufferTime value
        if (bufferTime < 0)
        {
            bufferTime = SoundMixer.bufferTime;
        }

        // Keeps buffer time reasonable, between 0 and 30 seconds
        this.bufferTime = Math.min(Math.max(0, bufferTime), 30000);

        if (autoLoad)
        {
            load();
        }
    }
```

La classe SoundFacade étend la classe EventDispatcher pour qu'elle puisse envoyer ses propres événements. Le code de classe déclare d'abord les propriétés pour un objet Sound et un objet SoundChannel. La classe stocke également la valeur de l'URL du fichier audio et une propriété `bufferTime` à utiliser lors de la lecture du son en continu. De plus, elle accepte des valeurs de paramètre booléennes qui affectent le comportement de lecture et de chargement :

- Le paramètre `autoLoad` indique à l'objet que le chargement du son doit commencer dès la création de cet objet.
- Le paramètre `autoPlay` indique que la lecture du son doit commencer dès qu'une quantité suffisante de données audio a été chargée. S'il s'agit d'un son diffusé en continu, la lecture commence dès qu'une quantité suffisante de données (comme spécifié par la propriété `bufferTime`) est chargée.
- Le paramètre `streaming` indique que ce fichier audio peut commencer la lecture avant la fin du chargement.

Le paramètre `bufferTime` prend la valeur -1 par défaut. Si la méthode constructeur détecte une valeur négative dans le paramètre `bufferTime`, elle définit la propriété `bufferTime` sur la valeur de `SoundMixer.bufferTime`. Ceci permet à l'application de prendre la valeur `SoundMixer.bufferTime` globale, par défaut, comme souhaité.

Si le paramètre `autoLoad` est défini sur `true`, la méthode constructeur appelle immédiatement la méthode `load()` suivante pour commencer le chargement du fichier audio:

```
public function load():void
{
    if (this.isPlaying)
    {
        this.stop();
        this.s.close();
    }
    this.isLoaded = false;

    this.s = new Sound();

    this.s.addEventListener(ProgressEvent.PROGRESS, onLoadProgress);
    this.s.addEventListener(Event.OPEN, onLoadOpen);
    this.s.addEventListener(Event.COMPLETE, onLoadComplete);
    this.s.addEventListener(Event.ID3, onID3);
    this.s.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
    this.s.addEventListener(SecurityErrorEvent.SECURITY_ERROR, onIOError);

    var req:URLRequest = new URLRequest(this.url);

    var context:SoundLoaderContext = new SoundLoaderContext(this.bufferTime, true);
    this.s.load(req, context);
}
```

La méthode `load()` crée un objet Sound puis ajoute des écouteurs pour tous les événements de son importants. Elle indique ensuite à l'objet Sound de charger le fichier audio, à l'aide d'un objet LoaderContext pour transmettre la valeur `bufferTime`.

Etant donné que la propriété `url` peut être modifiée, vous pouvez utiliser une occurrence de SoundFacade pour lire différents fichiers audio à la suite : il vous suffit de modifier la propriété `url` et d'appeler la méthode `load()` afin de charger le nouveau fichier audio.

Les trois méthodes d'écouteur d'événement suivantes indiquent comment l'objet SoundFacade suit la progression du chargement et décide quand lancer la lecture du son :

```
public function onLoadOpen(event:Event):void
{
    if (this.isStreaming)
    {
        this.isReadyToPlay = true;
        if (autoPlay)
        {
            this.play();
        }
    }
    this.dispatchEvent(event.clone());
}

public function onLoadProgress(event:ProgressEvent):void
{
    this.dispatchEvent(event.clone());
}

public function onLoadComplete(event:Event):void
{
    this.isReadyToPlay = true;
    this.isLoaded = true;
    this.dispatchEvent(evt.clone());

    if (autoPlay && !isPlaying)
    {
        play();
    }
}
```

La méthode `onLoadOpen()` s'exécute lorsque le chargement du son commence. Si vous pouvez lire le son en mode continu, la méthode `onLoadComplete()` définit immédiatement l'indicateur `isReadyToPlay` sur `true`. L'indicateur `isReadyToPlay` détermine si l'application peut lancer la lecture du son, peut-être en réponse à une action utilisateur (clic sur un bouton de lecture, par exemple). La classe SoundChannel gère la mise en mémoire tampon des données audio. Par conséquent, il est inutile de vérifier si suffisamment de données ont été chargées avant d'appeler la méthode `play()`.

La méthode `onLoadProgress()` s'exécute régulièrement pendant le chargement. Elle envoie simplement une copie de son objet ProgressEvent pour le code qui utilise cet objet SoundFacade.

Une fois que les données audio ont été complètement chargées, la méthode `onLoadComplete()` s'exécute en appelant la méthode `play()` pour des sons non diffusés en continu, si nécessaire. La méthode `play()` est décrite ci-dessous.

```
public function play(pos:int = 0):void
{
    if (!this.isPlaying)
    {
        if (this.isReadyToPlay)
        {
            this.sc = this.s.play(pos);
            this.sc.addEventListener(Event.SOUND_COMPLETE, onPlayComplete);
            this.isPlaying = true;

            this.playTimer = new Timer(this.progressInterval);
            this.playTimer.addEventListener(TimerEvent.TIMER, onPlayTimer);
            this.playTimer.start();
        }
    }
}
```

La méthode `play()` appelle la méthode `Sound.play()` lorsque le son peut être lu. L'objet SoundChannel résultant est stocké dans la propriété `sc`. La méthode `play()` crée ensuite un objet Timer qui sera utilisé pour envoyer des événements de progression de la lecture à des intervalles réguliers.

Affichage de la progression de la lecture

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La création d'un objet Timer pour surveiller la lecture est une opération complexe que vous devez coder une seule fois. Le fait d'encapsuler cette logique Timer dans une classe réutilisable telle que la classe SoundFacade permet aux applications d'écouter les mêmes types d'événements de progression lorsqu'un son est chargé et lorsqu'il est lu.

L'objet Timer créé par la méthode `SoundFacade.play()` envoie une occurrence de TimerEvent toutes les secondes. La méthode `onPlayTimer()` s'exécute chaque fois qu'un nouveau TimerEvent arrive :

```
public function onPlayTimer(event:TimerEvent):void
{
    var estimatedLength:int =
        Math.ceil(this.s.length / (this.s.bytesLoaded / this.s.bytesTotal));
    var progEvent:ProgressEvent =
        new ProgressEvent(PLAY_PROGRESS, false, false, this.sc.position, estimatedLength);
    this.dispatchEvent(progEvent);
}
```

La méthode `onPlayTimer()` implémente la technique d'estimation de la taille décrite dans la section « Surveillance de la lecture » à la page 467. Elle crée ensuite une occurrence de ProgressEvent avec un type d'événement de `SoundFacade.PLAY_PROGRESS`, avec la propriété `bytesLoaded` définie sur la position actuelle de l'objet SoundChannel et la propriété `bytesTotal` définie sur la longueur estimée des données audio.

Interruption et reprise de la lecture

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `SoundFacade.play()` décrite précédemment accepte un paramètre `pos` correspondant à une position de début dans les données audio. Si la valeur `pos` est zéro, la lecture du son commence au début.

La méthode `SoundFacade.stop()` accepte également un paramètre `pos`, comme indiqué ici :

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

Chaque fois que la méthode `SoundFacade.stop()` est appelée, elle définit la propriété `pausePosition` de façon à ce que l'application sache où positionner la tête de lecture si l'utilisateur souhaite reprendre la lecture du même son.

Les méthodes `SoundFacade.pause()` et `SoundFacade.resume()` indiquées ci-dessous appellent les méthodes `SoundFacade.stop()` et `SoundFacade.play()` respectivement, transmettant chaque fois une valeur de paramètre `pos`.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

La méthode `pause()` transmet la valeur `SoundChannel.position` actuelle à la méthode `play()`, qui la stocke dans la propriété `pausePosition`. La méthode `resume()` recommence à lire le même son en utilisant la valeur `pausePosition` comme point de début.

Extension de l'exemple Podcast Player

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple présente un Podcast Player dépouillé qui présente l'utilisation de la classe SoundFacade réutilisable. Vous pouvez ajouter d'autres fonctions pour améliorer l'utilité de cette application, notamment :

- stocker la liste des fils de syndication et des informations d'utilisation concernant chaque épisode dans une occurrence de SharedObject pouvant être utilisée la prochaine fois que l'utilisateur exécute l'application ;
- permettre à l'utilisateur d'ajouter son fil de syndication à la liste des chaînes de podcast ;
- mémoriser la position de la tête de lecture lorsque l'utilisateur arrête ou quitte un épisode de façon à ce qu'il puisse être redémarré à partir de ce point la prochaine fois que l'utilisateur exécute l'application ;
- télécharger des fichiers mp3 d'épisodes pour les écouter hors ligne, lorsque l'utilisateur n'est pas connecté à Internet ;
- ajouter des fonctions d'abonnement qui vérifient régulièrement la présence de nouveaux épisodes dans une chaîne de podcast et mettre à jour la liste des épisodes automatiquement ;
- ajouter une fonctionnalité de recherche de podcasts à l'aide d'une API à partir d'un service d'hébergement de podcasts, tel que Odeo.com.

Chapitre 25 : Utilisation de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La vidéo avec Flash est l'une des technologies dominantes d'Internet. Toutefois, l'interface traditionnelle de la vidéo, dans un écran rectangulaire avec une barre de progression surmontant des boutons de contrôle, n'est que l'un des usages possibles de la vidéo. En ActionScript, il est possible de contrôler avec précision le chargement, la présentation et la lecture d'une vidéo.

Principes de base de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La possibilité de lire et manipuler des informations vidéo en ActionScript, au même titre que les autres éléments multimédias (images, texte, animations, etc.) est l'une des principales caractéristiques d'Adobe® Flash® Player et d'Adobe® AIR™. Lorsque vous créez un fichier vidéo Flash (FLV) dans Adobe Flash CS4 Professional, vous avez la possibilité de sélectionner un habillage, ou « enveloppe », comportant les commandes de lecture courantes. Toutefois, vous n'êtes pas limité aux options disponibles. ActionScript offre un contrôle précis du chargement, de l'affichage et de la lecture de la vidéo, et vous pouvez créer votre propre enveloppe ou utiliser une vidéo de façon beaucoup moins traditionnelle. La gestion de la vidéo en ActionScript nécessite de travailler avec une combinaison de plusieurs classes :

- Classe Video : la zone de contenu vidéo traditionnelle affichée sur la scène est une occurrence de la classe Video. La classe Video est un objet d'affichage, qu'il est donc possible de manipuler à l'aide des mêmes techniques que les autres objets d'affichage (positionnement, application de transformations, de filtres et de modes de fusion, etc.).
- Classe StageVideo : la classe Video fait traditionnellement appel au décodage et au rendu logiciels. Si un périphérique gère l'accélération matérielle par processeur graphique, l'utilisation de la classe StageVideo permet à l'application d'exploiter pleinement la présentation à accélération matérielle. L'API StageVideo intègre un ensemble d'événements qui indiquent au code quand passer d'un objet StageVideo à un objet Video et inversement. La vidéo sur la scène impose diverses restrictions mineures en matière de lecture de vidéo. Si l'application gère ces restrictions, mettez en œuvre l'API StageVideo. Voir « Directives et restrictions » à la page 530.
- Classe NetStream : lorsque vous chargez un fichier vidéo qui doit être contrôlé en ActionScript, une occurrence de NetStream représente la source du contenu vidéo (dans ce cas précis, un flux de données vidéo). L'utilisation d'une occurrence de NetStream nécessite d'utiliser également un objet NetConnection, qui assure la connexion avec le fichier vidéo, comme un tunnel qu'emprunteraient les données vidéo.
- Classe Camera : si vous devez gérer des données provenant d'une caméra connectée à l'ordinateur de l'utilisateur, une occurrence de Camera représente la source du contenu vidéo (la caméra de l'utilisateur et les données vidéo qu'elle transmet). Nouveauté dans Flash Player 11.4 et AIR 3.4 : vous pouvez utiliser une caméra pour alimenter StageVideo.

Pour charger un fichier vidéo externe, vous pouvez charger ce fichier à partir d'un serveur Web standard (téléchargement progressif) ou gérer de la vidéo en flux continu transmise par un serveur spécialisé tel que Flash® Media Server d'Adobe.

Concepts importants et terminologie

Point de repère Marqueur qu'il est possible de placer en un point spécifique d'un fichier vidéo, notamment pour l'utiliser comme signet pour repérer ce point à partir du début de la vidéo ou pour fournir des données supplémentaires associées à ce moment de la vidéo.

Codage Processus de conversion de données vidéo d'un format dans un autre format vidéo. Par exemple, la conversion d'une vidéo source en haute résolution dans un format plus adapté à la diffusion sur Internet.

Image Elément de base des informations vidéo. Chaque image s'apparente à un cliché photographique représentant un moment précis. La lecture en séquence à vitesse élevée de ces images fixes donne l'illusion du mouvement.

Image-clé Image vidéo qui contient l'ensemble des informations de l'image. Les autres images qui suivent une image-clé ne contiennent que les informations décrivant leurs différences par rapport à l'image-clé, et non pas les informations d'image complètes.

Métadonnées Informations sur un fichier vidéo intégrées à ce fichier et lues après son chargement.

Téléchargement progressif Lorsqu'un fichier vidéo est transmis par un serveur Web standard, les données vidéo sont chargées en mode progressif, c'est-à-dire en séquences. L'avantage est qu'il est possible de commencer à diffuser la vidéo avant la fin du téléchargement complet. Toutefois, il est alors impossible de passer directement à une partie de la vidéo qui n'a pas encore été chargée.

Diffusion en continu Pour éviter le téléchargement progressif, il est possible d'utiliser un serveur vidéo spécial pour diffuser de la vidéo sur Internet selon une technique connue sous le nom de diffusion en continu. Avec la diffusion en flux continu, l'ordinateur client ne charge jamais toute la vidéo à la fois. Pour accélérer les délais de chargement, l'ordinateur n'a besoin, à un moment donné, que d'une partie de l'ensemble des informations vidéo. Comme un serveur spécial contrôle la diffusion du contenu vidéo, une partie quelconque de celle-ci peut être transmise à tout moment, et il n'est donc pas nécessaire d'attendre qu'elle soit chargée pour y accéder.

Présentation des formats vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Outre le format vidéo Adobe FLV, Flash Player et Adobe AIR prennent en charge les contenus vidéo et audio codés au format H.264 et HE-AAC des formats de fichier standard MPEG-4. Ces formats diffusent des vidéos de qualité supérieure à des débits inférieurs. Les développeurs peuvent utiliser les outils standard, notamment Adobe Premiere Pro et Adobe After Effects, pour créer et présenter du contenu vidéo de grande qualité.

Type	Format	Contenant
Vidéo	H.264	MPEG-4 : MP4, M4V, F4V, 3GPP
Vidéo	Sorenson Spark	Fichier FLV
Vidéo	ON2 VP6	Fichier FLV
Audio	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4:MP4, M4V, F4V, 3GPP
Audio	Mp3	Mp3
Audio	Nellymoser	Fichier FLV
Audio	Speex	Fichier FLV

Voir aussi

Flash Media Server : codecs pris en charge

Technologie HTTP Dynamic Streaming d'Adobe

Codage vidéo destiné aux périphériques mobiles

AIR sur Android peut décoder un large éventail de vidéos H.264. Toutefois, seul un jeu partiel réduit de vidéos H.264 bénéficie d'une lecture fluide sur un téléphone portable. De nombreux téléphones portables ne disposent en effet pas d'une puissance de traitement suffisante. Adobe Flash Player pour périphérique mobile peut décoder les vidéos H.264 par le biais d'une accélération matérielle intégrée. Ce décodage allie une qualité de lecture supérieure à une consommation réduite.

Le standard H.264 prend en charge plusieurs techniques d'encodage. Seuls les périphériques de pointe assurent une lecture fluide avec des profils et niveaux complexes. La plupart des périphériques peut toutefois lire des vidéos codées en profil de base. Sur les périphériques mobiles, un jeu partiel de ces techniques exploite l'accélération matérielle. Le profil et les paramètres de niveau définissent ce jeu partiel de paramètres et techniques de codage utilisé par l'encodeur. Pour les développeurs, il se traduit par la conversion de la vidéo à la résolution sélectionnée, assurant ainsi une lecture fluide sur la plupart des périphériques.

Bien que les résolutions qui exploitent l'accélération matérielle varient d'un périphérique à l'autre, la plupart de ces derniers prend en charge les résolutions standard suivantes.

Format	**Résolutions recommandées**		
4:3	640 × 480	512 × 384	480 × 360
16:9	640 × 360	512 x 288	480 × 272

Remarque : *Flash Player prend en charge tous les niveaux et profils du standard H.264. Le respect de ces recommandations assure l'accélération matérielle et une expérience utilisateur optimisée sur la plupart des périphériques. Ces recommandations ne sont pas obligatoires.*

Pour plus d'informations et pour obtenir la liste des paramètres d'encodage dans Adobe Media Encoder CS5, voir Recommendations for encoding H.264 video for Flash Player 10.1 on mobile devices (disponible en anglais uniquement).

Remarque : *sous iOS, seules les vidéos codées avec les codecs Sorenson Spark et On2 VP6 peuvent être lues à l'aide de la classe Video. Vous pouvez lire les données vidéos H.264 dans le lecteur vidéo du périphérique en accédant à l'URL de la vidéo à l'aide de la fonction* `flash.net.navigateToURL()`*. Vous pouvez également lire les données vidéo H.264 à l'aide de la balise <video> sur une page html affichée dans un objet StageWebView.*

Compatibilité de Flash Player et AIR avec les fichiers vidéo codés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player 7 prend en charge les fichiers FLV codés avec le codec vidéo Sorenson™ Spark™. Flash Player 8 prend en charge les fichiers FLV codés avec l'encodeur Sorenson ou On2 VP6 dans Flash Professional 8. Le codec On2 VP6 prend en charge un canal alpha.

Flash Player 9.0.115.0 et les versions ultérieures prennent en charge les fichiers dérivés du format conteneur standard MPEG-4. Ces fichiers sont les suivants : F4V, MP4, M4A, MOV, MP4V, 3GP et 3G2, s'ils contiennent de la vidéo H.264 ou de l'audio codé au format HE-AAC v2, ou les deux. H.264 produit une qualité vidéo supérieure à un débit inférieur par rapport au même profil d'encodage dans Sorenson ou On2. HE-AAC v2 est une extension du format AAC, format audio standard défini dans la norme vidéo MPEG-4. HE-AAC v2 utilise les techniques de réplique spectrale de bande (SBR - Spectral Band Replication) et de stéréo paramétrique pour optimiser l'efficacité de l'encodage à des vitesses de transfert inférieures.

Le tableau suivant répertorie les codecs pris en charge. Il décrit également le format de fichier SWF correspondant et les versions de Flash Player et d'AIR requises pour les lire :

Codec	**Version du format de fichier SWF (version de publication la plus récente prise en charge)**	**Flash Player et AIR (version la plus récente requise pour la lecture)**
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3 Seuls Flash Player 8 et les versions ultérieures prennent en charge la publication et la lecture des vidéos On2 VP6.
H.264 (MPEG-4 Part 10)	9	Flash Player 9 Mise à jour 3, AIR 1.0
ADPCM	6	Flash Player 6, Flash Lite 3
Mp3	6	Flash Player 6, Flash Lite 3
AAC (MPEG-4 Part 3)	9	Flash Player 9 Mise à jour 3, AIR 1.0
Speex (audio)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

Présentation des formats de fichiers vidéo Adobe F4V et FLV

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Adobe fournit des formats de fichiers vidéo pour diffuser en continu un contenu à Flash Player et à AIR. Pour une description complète de ces formats de fichiers vidéo, voir www.adobe.com/go/video_file_format_fr.

Format de fichier vidéo F4V

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

A partir de Flash Player Update 3 (9.0.115.0) et AIR 1.0, Flash Player et AIR prennent en charge le format vidéo Adobe F4V qui découle du format ISO MP4. Les sous-ensembles du format prennent en charge des fonctions diverses. Flash Player s'attend à ce qu'un fichier F4V valide commence par l'une des boîtes de haut niveau suivantes :

- ftyp

  La boîte ftyp identifie les fonctions qu'un programme doit prendre en charge pour diffuser un format de fichier particulier.

- moov

  La boîte moov est effectivement l'en-tête d'un fichier F4V. Elle contient une ou plusieurs autres boîtes qui à leur tour contiennent d'autres boîtes et qui définissent la structure des données F4V. Un fichier F4V ne doit contenir qu'une seule boîte moov.

- mdat

  Une boîte mdat contient les données " utiles " pour le fichier F4V. Un fichier F4V ne doit contenir qu'une seule boîte mdat. Une boîte moov doit également se trouver dans le fichier parce que l boîte mdat ne peut pas être comprise si elle est seule.

Les fichiers F4V prennent en charge des entiers multioctets dans un ordre d'octets gros-boutiste, dans lequel l'octet le plus significatif paraît le premier, à l'adresse la plus basse.

Format de fichier vidéo FLV

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le format de fichier Adobe FLV contient des données audio et vidéo codées pour un acheminement via Flash Player. Vous pouvez utiliser un codeur, tel qu'Adobe Media Encoder ou Sorenson™ Squeeze, pour convertir un fichier vidéo QuickTime ou Windows Media en un fichier FLV.

Remarque : *vous pouvez créer des fichiers FLV en important la vidéo dans Flash et en l'exportant sous forme de fichier FLV. Vous pouvez utiliser le module d'exportation FLV pour exporter des fichiers FLV à partir des applications de montage vidéo prises en charge. Pour charger des fichiers FLV à partir d'un serveur Web, enregistrez l'extension de fichier et le type MIME auprès de votre serveur Web. Pour ce faire, consultez la documentation du serveur. Le type MIME des fichiers FLV est* `video/x-flv`*. Pour plus d'informations, voir «* *A propos de la configuration de fichier FLV pour l'hébergement sur un serveur » à la page 522.*

Pour plus d'informations sur les fichiers FLV, voir « Rubriques avancées relatives aux fichiers vidéo » à la page 521.

Vidéo externe contre vidéo intégrée

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'utilisation des fichiers vidéo externes offre certaines fonctionnalités qui ne sont pas disponibles avec l'utilisation de la vidéo importée :

- Les clips vidéo de longue durée peuvent être utilisés dans votre application sans ralentir la lecture. Les fichiers vidéo externes utilisent la mémoire cache, ce qui signifie que les fichiers volumineux sont enregistrés en petites parties et sont accessibles dynamiquement. Par conséquent, les fichiers F4V et FLV externes ne nécessitent pas autant de mémoire que les fichiers vidéo intégrés.
- Un fichier vidéo externe peut avoir une cadence différente de celle du fichier SWF dans lequel il est lu. Par exemple, vous pouvez définir la cadence du fichier SWF sur 30i/s (images par seconde) et celle de l'image vidéo sur 21i/s. Ce réglage vous offre un meilleur contrôle de la vidéo que la vidéo intégrée, pour assurer une lecture vidéo fluide. Il permet aussi de lire les fichiers vidéo à différentes cadences d'images sans avoir à altérer un contenu SWF existant.
- Avec les fichiers vidéo externes, la lecture du contenu SWF n'est pas interrompue lors du chargement du fichier vidéo. Les fichiers vidéo importés peuvent parfois interrompre la lecture du document pour exécuter certaines fonctions, par exemple pour accéder à un lecteur de CD-ROM. Les fichiers vidéo peuvent exécuter des fonctions indépendamment du contenu SWF, sans interrompre la lecture.
- Le sous-titrage du contenu vidéo est plus facile avec les fichiers FLV externes, car les gestionnaires d'événements permettent d'accéder aux métadonnées de la vidéo.

Présentation de la classe Video

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Video permet d'afficher un flux vidéo en direct dans une application sans l'imbriquer dans votre fichier SWF. Vous pouvez capturer et lire du contenu vidéo en direct à l'aide de la méthode `Camera.getCamera()`. Vous pouvez également utiliser la classe Video pour lire les fichiers vidéo en HTTP ou sur le système de fichiers local. Il existe plusieurs façons d'utiliser la classe Video dans vos projets.

- Chargement dynamique d'un fichier vidéo à l'aide des classes NetConnection et NetStream, et affichage de la vidéo dans un objet Video.
- Capture du signal provenant de la caméra de l'utilisateur. Pour plus d'informations, voir « Utilisation de caméras » à la page 537.
- Utilisation du composant FLVPlayback.
- Utilisation de la commande VideoDisplay.

Remarque : *les occurrences d'un objet Video sur la scène sont des occurrences de la classe Video.*

Bien que la classe Video se trouve dans le package flash.media, elle hérite de la classe flash.display.DisplayObject. Par conséquent, toutes les fonctionnalités des objets d'affichage (transformation de matrice et filtres par exemple) s'appliquent aussi aux occurrences de l'objet Video.

Pour plus d'informations, voir « Manipulation des objets d'affichage » à la page 180, ainsi que les chapitres « Utilisation de la géométrie » à la page 218 et « Filtrage des objets d'affichage » à la page 276.

Chargement de fichiers vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le chargement de fichiers vidéo à l'aide des classes NetStream et NetConnection s'effectue en plusieurs étapes. Il est recommandé d'ajouter l'objet Video à la liste d'affichage, de joindre l'objet NetStream à l'occurrence de l'objet Video et d'appeler la méthode `play()` de l'objet NetStream dans l'ordre spécifié :

1. Créez un objet NetConnection. Dans le cas d'une connexion à un fichier vidéo local ou à un fichier qui n'utilise pas de serveur, tel que le serveur Flash Media Server 2 d'Adobe, transmettez `null` à la méthode `connect()` pour lire les fichiers vidéo depuis une adresse HTTP ou un lecteur local. Dans le cas d'une connexion à un serveur, définissez le paramètre sur l'URI de l'application qui contient le fichier vidéo sur le serveur.

   ```
   var nc:NetConnection = new NetConnection();
   nc.connect(null);
   ```

2. Créez un nouvel objet Video qui affiche la vidéo et ajoutez-le à la liste d'affichage sur la scène, comme indiqué dans l'extrait de code suivant :

   ```
   var vid:Video = new Video();
   addChild(vid);
   ```

3. Créez un objet NetStream en transmettant l'objet NetConnection au constructeur en tant qu'argument. L'extrait de code suivant connecte un objet NetStream à une occurrence de NetConnection et configure les gestionnaires d'événement pour le flux de données :

```
var ns:NetStream = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS,netStatusHandler);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);

function netStatusHandler(event:NetStatusEvent):void
{
    // handle netStatus events, described later
}

function asyncErrorHandler(event:AsyncErrorEvent):void
{
    // ignore error
}
```

4 Joignez l'objet NetStream à l'objet Video à l'aide de la méthode `attachNetStream()` de l'objet Video, comme indiqué dans l'extrait de code suivant :

```
vid.attachNetStream(ns);
```

5 Appelez la méthode `play()` de l'objet NetStream avec l'URL du fichier vidéo comme argument pour lancer la lecture de la vidéo. L'extrait de code suivant charge et lit un fichier vidéo appelé « video.mp4 » dans le même répertoire que le fichier SWF :

```
ns.play("video.mp4");
```

Voir aussi

Flex : Contrôle Spark VideoPlayer

spark.components.VideoDisplay

Contrôle de la lecture de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe NetStream comporte quatre méthodes principales pour contrôler la lecture vidéo :

`pause()` : interrompt la lecture d'un flux vidéo. Si la lecture de la vidéo est déjà interrompue, l'appel de cette méthode n'a aucun effet.

`resume()` : reprend la lecture d'un flux vidéo interrompu. Si la vidéo est en cours de lecture, l'appel de cette méthode n'a aucun effet.

`seek()` : recherche l'image-clé la plus proche de l'emplacement spécifié (décalage, exprimé en secondes, par rapport au début du flux).

`togglePause()` : interrompt ou reprend la lecture d'un flux.

Remarque : *il n'existe pas de méthode `stop()`. Pour arrêter la lecture de la vidéo, il est nécessaire de la mettre en pause et de retourner au début du flux vidéo.*

Remarque : *la méthode `play()` ne reprend pas la lecture, elle est destinée au chargement de fichiers vidéo.*

L'exemple suivant montre comment contrôler la lecture d'une vidéo à l'aide de divers boutons. Pour exécuter cet exemple, créez un document et ajoutez quatre occurrences de boutons à l'espace de travail (`pauseBtn`, `playBtn`, `stopBtn` et `togglePauseBtn`) :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);
ns.play("video.flv");
function asyncErrorHandler(event:AsyncErrorEvent):void
{
    // ignore error
}

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

pauseBtn.addEventListener(MouseEvent.CLICK, pauseHandler);
playBtn.addEventListener(MouseEvent.CLICK, playHandler);
stopBtn.addEventListener(MouseEvent.CLICK, stopHandler);
togglePauseBtn.addEventListener(MouseEvent.CLICK, togglePauseHandler);

function pauseHandler(event:MouseEvent):void
{
    ns.pause();
}
function playHandler(event:MouseEvent):void
{
    ns.resume();
}
function stopHandler(event:MouseEvent):void
{
    // Pause the stream and move the playhead back to
    // the beginning of the stream.
    ns.pause();
    ns.seek(0);
}
function togglePauseHandler(event:MouseEvent):void
{
    ns.togglePause();
}
```

Un clic sur l'occurrence de bouton `pauseBtn` pendant la lecture de la vidéo provoque la mise en pause de celle-ci. Si la lecture de la vidéo est déjà en pause, l'appel de cette méthode n'a aucun effet. Un clic sur l'occurrence de `playBtn` reprend la lecture de la vidéo si celle-ci était en pause, sinon ce bouton n'a aucun effet.

Détection de la fin d'un flux vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour afficher le début et la fin d'un flux vidéo, vous devez ajouter à l'occurrence de NetStream un écouteur pour l'événement `netStatus`. L'exemple suivant montre comment écouter les divers codes pendant la lecture de la vidéo :

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

Le code précédent affiche le résultat suivant :

```
NetStream.Play.Start
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Empty
NetStream.Buffer.Full
NetStream.Buffer.Flush
NetStream.Play.Stop
NetStream.Buffer.Empty
NetStream.Buffer.Flush
```

Les deux codes d'événement qu'il est nécessaire d'écouter sont « NetStream.Play.Start » et « NetStream.Play.Stop », qui signalent le début et la fin de la lecture de la vidéo. Le fragment de code suivant utilise une instruction « switch » pour filtrer ces deux codes et émettre un message :

```
function statusHandler(event:NetStatusEvent):void
{
    switch (event.info.code)
    {
        case "NetStream.Play.Start":
            trace("Start [" + ns.time.toFixed(3) + " seconds]");
            break;
        case "NetStream.Play.Stop":
            trace("Stop [" + ns.time.toFixed(3) + " seconds]");
            break;
    }
}
```

En écoutant l'événement `netStatus` (`NetStatusEvent.NET_STATUS`), vous pouvez créer un lecteur vidéo qui chargera la vidéo suivante dans une liste de lecture une fois la lecture de la vidéo en cours terminée.

Lecture de vidéos en mode plein écran

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player et AIR vous permettent de créer une application plein écran pour la lecture de votre vidéo et prennent en charge la mise à l'échelle de la vidéo au mode plein écran.

Pour le contenu AIR qui s'exécute en mode plein écran sur le poste de travail, les options économiseur d'écran et économie d'énergie du système sont désactivées lors de la lecture jusqu'à ce que le signal vidéo s'arrête ou que l'utilisateur quitte le mode plein écran.

Pour plus d'informations sur l'utilisation du mode plein écran, voir « Utilisation du mode Plein écran » à la page 173.

Activation du mode plein écran pour Flash Player dans un navigateur

Avant que vous ne puissiez implémenter le plein écran pour Flash Player dans un navigateur, activez-le via le modèle de publication de votre application. Les modèles qui permettent le mode plein écran comprennent les balises `<object>` et `<embed>`, qui contiennent un paramètre `allowFullScreen`. L'exemple suivant affiche le paramètre `allowFullScreen` dans une balise `<embed>`.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    id="fullScreen" width="100%" height="100%"
    codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab">
    ...
    <param name="allowFullScreen" value="true" />
    <embed src="fullScreen.swf" allowFullScreen="true" quality="high" bgcolor="#869ca7"
        width="100%" height="100%" name="fullScreen" align="middle"
        play="true"
        loop="false"
        quality="high"
        allowScriptAccess="sameDomain"
        type="application/x-shockwave-flash"
        pluginspage="http://www.adobe.com/go/getflashplayer">
    </embed>
    ...
</object>
```

Dans Flash, choisissez Fichier -> Paramètres de publication, puis dans la boîte de dialogue Paramètres de publication, cliquez sur l'onglet HTML, puis sélectionnez le modèle Flash seulement - Autorisation du Plein écran.

Dans Flex, assurez-vous que le modèle HTML inclut les balises `<object>` et `<embed>` qui prennent en charge le plein écran.

Activation du mode plein écran

Pour le contenu de Flash Player s'exécutant dans un navigateur, le mode plein écran de la vidéo est activé en réponse à un clic de souris ou à une pression sur une touche. Par exemple, vous pouvez activer le mode plein écran lorsque l'utilisateur clique sur un bouton appelé Plein écran ou sélectionne une commande Plein écran dans un menu contextuel. Pour répondre à l'utilisateur, ajoutez un écouteur d'événement à l'objet sur lequel se produit l'action. Le code suivant ajoute un écouteur d'événement à un bouton sur lequel l'utilisateur clique pour accéder au mode plein écran :

```
var fullScreenButton:Button = new Button();
fullScreenButton.label = "Full Screen";
addChild(fullScreenButton);
fullScreenButton.addEventListener(MouseEvent.CLICK, fullScreenButtonHandler);

function fullScreenButtonHandler(event:MouseEvent)
{
    stage.displayState = StageDisplayState.FULL_SCREEN;

}
```

Le code active le mode plein écran en définissant la propriété `Stage.displayState` sur `StageDisplayState.FULL_SCREEN`. Ce code affiche la totalité de la scène en mode plein écran et met à l'échelle la vidéo selon les proportions de l'espace qu'elle occupe sur la scène.

La propriété `fullScreenSourceRect` vous permet de spécifier une zone particulière de la scène en vue de l'afficher en mode plein écran. Définissez tout d'abord le rectangle que vous souhaitez afficher en mode plein écran. Ensuite, affectez-le à la propriété `Stage.fullScreenSourceRect`. Cette version de la fonction `fullScreenButtonHandler()` ajoute deux lignes supplémentaires de code qui n'activent le plein écran que pour la vidéo.

```
private function fullScreenButtonHandler(event:MouseEvent)
{
    var screenRectangle:Rectangle = new Rectangle(video.x, video.y, video.width, video.height);
    stage.fullScreenSourceRect = screenRectangle;
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Bien que cet exemple invoque un gestionnaire d'événement en réponse à un clic de souris, la technique d'activation du mode plein écran est la même pour Flash Player et pour AIR. Définissez le rectangle que vous souhaitez mettre à l'échelle, puis définissez la propriété `Stage.displayState`. Pour plus d'informations, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

L'exemple complet qui suit ajoute le code permettent de créer la connexion et l'objet NetStream pour la vidéo, puis commence à le lire.

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.media.Video;
    import flash.display.StageDisplayState;
    import fl.controls.Button;
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.events.FullScreenEvent;
    import flash.geom.Rectangle;

    public class FullScreenVideoExample extends Sprite
    {
        var fullScreenButton:Button = new Button();
        var video:Video = new Video();

        public function FullScreenVideoExample()
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;

            addChild(video);

            video.attachNetStream(videoStream);

            videoStream.play("http://www.helpexamples.com/flash/video/water.flv");

            fullScreenButton.x = 100;
```

```
            fullScreenButton.y = 270;
            fullScreenButton.label = "Full Screen";
            addChild(fullScreenButton);
            fullScreenButton.addEventListener(MouseEvent.CLICK, fullScreenButtonHandler);
        }

        private function fullScreenButtonHandler(event:MouseEvent)
        {
            var screenRectangle:Rectangle = new Rectangle(video.x, video.y, video.width,
video.height);
            stage.fullScreenSourceRect = screenRectangle;
            stage.displayState = StageDisplayState.FULL_SCREEN;
        }

        public function onMetaData(infoObject:Object):void
        {
            // stub for callback function
        }
    }
}
```

La fonction `onMetaData()` est une fonction de rappel pour gérer les métadonnées de la vidéo, si celles-ci existent. Une fonction de rappel est une fonction que le moteur d'exécution appelle en réponse à certains types d'occurrences ou d'événements. Dans cet exemple, la fonction `onMetaData()` est une souche capable de fournir la fonction. Pour plus d'informations, voir « Ecriture de méthodes de rappel pour les métadonnées et les points de repère » à la page 502

Désactivation du mode plein écran

Tout utilisateur peut désactiver le mode plein écran à l'aide de l'un des raccourcis clavier, notamment en appuyant sur la touche Echap. En ActionScript, vous pouvez désactiver le mode plein écran en définissant la propriété `Stage.displayState` sur `StageDisplayState.NORMAL`. Dans l'exemple suivant, le code désactive le mode plein écran lorsque l'événement `netStatus` de NetStream.Play.Stop se produit.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

Accélération matérielle en mode plein écran

Lorsque vous modifiez la mise à l'échelle d'une zone rectangulaire de la scène en lui appliquant le mode Plein écran, Flash Player ou AIR utilise l'accélération matérielle, à condition que cette fonction soit disponible et activée. Le moteur d'exécution utilise l'adaptateur vidéo de l'ordinateur pour accélérer la mise à l'échelle de la vidéo ou d'une partie de la scène au mode Plein écran. Dans ces cas, les applications Flash Player peuvent souvent en profiter en basculant sur la classe StageVideo à partir de la classe Video (ou de la classe Camera pour Flash Player 11.4/AIR 3.4 et les versions ultérieures).

Pour plus d'informations sur l'accélération matérielle en mode Plein écran, voir « Utilisation du mode Plein écran » à la page 173. Pour plus d'informations sur StageVideo, voir « Présentation à accélération matérielle par le biais de la classe StageVideo » à la page 528.

Lecture de fichiers vidéo en flux continu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour effectuer une lecture en flux continu à partir d'un serveur Flash Media Server, vous pouvez utiliser les classes NetConnection et NetStream afin d'établir la connexion avec une occurrence de serveur distant et lire le flux spécifié. Pour spécifier un serveur RTMP (Real-Time Messaging Protocol), il suffit de transmettre l'URL RTMP désirée, par exemple « `rtmp://localhost/appName/appInstance` » à la méthode `NetConnection.connect()` au lieu de lui transmettre la valeur null. Pour lire un flux vidéo direct ou enregistré à partir du serveur Flash Media Server spécifié, passez soit un identifiant (pour le signal vidéo en direct publié par `NetStream.publish()`), soit le nom du fichier enregistré, à la méthode `NetStream.play()`.

Envoi de vidéo à un serveur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous souhaitez créer des applications plus complexes avec des objets Video ou Camera, Flash Media Server (FMS) offre diverses possibilités de diffusion de flux multimédia ainsi qu'un environnement de développement pour créer des applications multimédia et les distribuer à l'intention d'un public très large. Cette combinaison permet aux développeurs de créer des applications telles que vidéo à la demande, diffusion d'événements en direct sur le Web et diffusion en flux continu MP3, mais aussi blog vidéo, vidéomessagerie ou conversation multimédia. Pour plus d'informations, voir la documentation de Flash Media Server disponible en ligne à l'adresse suivante : www.adobe.com/go/learn_fms_docs_fr.

Présentation des points de repère

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez intégrer des points de repère dans un fichier vidéo F4V ou FLV durant le codage. A l'origine, les points de repère étaient intégrés dans des films pour signaler visuellement au projectionniste que la bobine s'approchait de la fin. Dans les formats vidéo Adobe F4V et FLV, un point de repère vous permet de déclencher une ou plusieurs actions dans votre application au moment où il survient dans le flux vidéo.

Vous pouvez utiliser plusieurs types de points de repère avec Flash Video. ActionScript permet d'interagir avec les points de repère que vous intégrez dans un fichier vidéo lorsque vous le créez.

- Points de repère de navigation : vous intégrez des points de repère de navigation dans le flux et le paquet de métadonnées vidéo lorsque vous codez le fichier vidéo. Les points de repère de navigation permettent aux utilisateurs de rechercher une partie spécifique d'un fichier.
- Points de repère d'événement : vous intégrez des points de repère d'événement dans le flux et le paquet de métadonnées vidéo lorsque vous codez le fichier vidéo. Il est possible d'écrire du code pour gérer les événements qui sont déclenchés aux points spécifiés pendant la lecture.

- Points de repère ActionScript : les points de repère ActionScript sont disponibles uniquement pour le composant FLVPlayback de Flash. Les points de repère ActionScript sont des points de repère externes que vous créez et auxquels vous accédez à l'aide du code ActionScript. Du code permet de déclencher ces points de repère en fonction de la lecture vidéo. Ces points de repère sont moins précis que les points de repère intégrés (jusqu'à un dixième de seconde), car le lecteur vidéo les analyse séparément. Si vous avez l'intention de créer une application dans laquelle il sera possible d'atteindre un point de repère, créez et intégrez les points de repère lors de l'encodage du fichier, au lieu d'utiliser des points de repère ActionScript. Il est préférable d'intégrer les points de repère dans le fichier FLV, car ils sont alors plus précis.

Les points de repère de navigation créent une image-clé à l'emplacement spécifié, pour permettre de déplacer la tête de lecture du lecteur vidéo à cet emplacement. Vous pouvez définir des points particuliers dans un fichier vidéo pour permettre aux utilisateurs d'atteindre un emplacement précis. Par exemple, si votre vidéo contient plusieurs chapitres et segments, vous pouvez la contrôler en intégrant des points de repère de navigation dans le fichier vidéo.

Pour plus d'informations sur le codage de fichiers vidéo Adobe avec des points de repère, voir « Intégration de points de repère » dans *Utilisation de Flash*.

Vous pouvez accéder aux paramètres des points de repère à l'aide de code ActionScript. Les paramètres de points de repère font partie de l'objet d'événement reçu du gestionnaire de rappel.

Le gestionnaire d'événement `NetStream.onCuePoint` permet de déclencher certaines actions dans votre code lorsque le fichier FLV atteint un point de repère spécifique.

Pour synchroniser une action avec un point de repère dans un fichier vidéo F4V, vous devez récupérer les données de point de repère des fonctions de rappel `onMetaData()` ou `onXMPData()` et déclencher le point de repère à l'aide de la classe Timer dans ActionScript 3.0. Pour plus d'informations sur les points de repère de F4V, voir « Utilisation d'onXMPData() » à la page 514.

Pour plus d'informations sur l'utilisation des points de repère et des métadonnées, voir « Ecriture de méthodes de rappel pour les métadonnées et les points de repère » à la page 502.

Ecriture de méthodes de rappel pour les métadonnées et les points de repère

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez déclencher des actions au sein de votre application lorsque le lecteur reçoit des métadonnées spécifiques ou lorsque des points de repère particuliers sont atteints. Lorsque ces événements se produisent, vous devez utiliser des méthodes de rappel spécifiques en tant que gestionnaires d'événements. La classe NetStream spécifie les événements de métadonnées suivants qui se produisent lors de la lecture : `onCuePoint` (fichiers FLV uniquement), `onImageData`, `onMetaData`, `onPlayStatus`, `onTextData` et `onXMPData`.

Si vous ne créez pas de méthodes de rappel pour ces gestionnaires, le moteur d'exécution Flash risque de générer des erreurs. Par exemple, le code ci-dessous lit le fichier FLV video.flv situé dans le même dossier que le fichier SWF :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);
ns.play("video.flv");
function asyncErrorHandler(event:AsyncErrorEvent):void
{
    trace(event.text);
}

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Le code ci-dessus charge un fichier vidéo local nommé video.flv et attend la distribution de l'événement `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`). Cet événement est distribué lorsqu'une exception est renvoyée par du code asynchrone natif. Dans notre cas, il est distribué lorsque le fichier vidéo contient des métadonnées ou des informations de point de repère et que les écouteurs appropriés n'ont pas été définis. Le code ci-dessus gère l'événement `asyncError` et ignore l'erreur si vous n'êtes pas intéressé par les métadonnées ou les informations de point de repère. Si vous disposiez d'un fichier FLV avec des métadonnées et plusieurs points de repère, la fonction `trace()` afficherait les messages d'erreur suivants :

```
Error #2095: flash.net.NetStream was unable to invoke callback onMetaData.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
Error #2095: flash.net.NetStream was unable to invoke callback onCuePoint.
```

L'erreur est renvoyée parce que l'objet NetStream n'a pas trouvé de méthode de rappel pour `onMetaData` ou `onCuePoint`. Il existe plusieurs façons de définir ces méthodes de rappel dans une application.

Voir aussi

Flash Media Server : Gestion des métadonnées dans les flux

Définir la propriété « client » de l'objet NetStream comme Object

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En définissant la propriété `client` comme étant soit un object, soit une sous-classe de NetStream, vous pouvez re-router les méthodes de rappel de `onMetaData` et `onCuePoint` ou les ignorer totalement. L'exemple suivant montre comment utiliser un objet vide pour ignorer les méthodes de rappel sans écouter l'événement `asyncError` :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var customClient:Object = new Object();

var ns:NetStream = new NetStream(nc);
ns.client = customClient;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Pour écouter les méthodes de rappel de `onMetaData` ou `onCuePoint`, il est nécessaire de définir des méthodes pour les gérer, comme dans le fragment de code suivant :

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

Le code ci-dessus écoute la méthode de rappel de `onMetaData` et appelle la méthode `metaDataHandler()`, qui renvoie une chaîne. Si le moteur d'exécution Flash trouve un point de repère, aucune erreur n'est renvoyée, bien qu'aucune méthode de rappel de `onCuePoint` ne soit définie.

Créer une classe et définir des méthodes pour gérer les méthodes de rappel

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le code suivant définit la propriété `client` de l'objet NetStream comme étant une classe personnalisée, CustomClient, qui définit des gestionnaires pour les méthodes de rappel :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = new CustomClient();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

La classe CustomClient contient le code suivant :

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

La classe CustomClient définit un gestionnaire pour le rappel de `onMetaData`. Si un point de repère est détecté et que le gestionnaire de rappel de `onCuePoint` est appelé, un événement `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) est distribué pour informer que flash.net.NetStream n'a pas pu appeler de rappel pour onCuePoint. Pour éviter l'apparition de cette erreur, il est nécessaire de définir soit une méthode de rappel de `onCuePoint` dans la classe CustomClient, soit un gestionnaire d'événement pour l'événement `asyncError`.

Etendre la classe NetStream et lui ajouter des méthodes pour gérer les méthodes de rappel

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le code suivant crée une occurrence de la classe CustomNetStream, qui sera définie ultérieurement :

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

L'exemple de code suivant définit la classe CustomNetStream qui étend la classe NetStream, prend en charge la création de l'objet NetConnection nécessaire, et gère les méthodes de rappel de `onMetaData` et `onCuePoint` :

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public class CustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public function CustomNetStream()
        {
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
        public function onCuePoint(infoObject:Object):void
        {
            trace("cue point");
        }
    }
}
```

Si vous souhaitez renommer les méthodes `onMetaData()` et `onCuePoint()` de la classe CustomNetStream, utilisez le code suivant :

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public class CustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public var onMetaData:Function;
        public var onCuePoint:Function;
        public function CustomNetStream()
        {
            onMetaData = metaDataHandler;
            onCuePoint = cuePointHandler;
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
        private function metaDataHandler(infoObject:Object):void
        {
            trace("metadata");
        }
        private function cuePointHandler(infoObject:Object):void
        {
            trace("cue point");
        }
    }
}
```

Etendre la classe NetStream et la rendre dynamique

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez étendre la classe NetStream en créant une sous-classe dynamique afin de pouvoir ajouter dynamiquement les gestionnaires de rappel de `onCuePoint` et `onMetaData`. Le code suivant en fait la démonstration :

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

La classe DynamicCustomNetStream contient le code suivant :

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    public dynamic class DynamicCustomNetStream extends NetStream
    {
        private var nc:NetConnection;
        public function DynamicCustomNetStream()
        {
            nc = new NetConnection();
            nc.connect(null);
            super(nc);
        }
    }
}
```

Même sans gestionnaire pour le rappel de `onMetaData` et `onCuePoint`, aucune erreur n'est renvoyée puisque la classe DynamicCustomNetStream est dynamique. Si vous souhaitez définir des méthodes pour les gestionnaires de rappel de `onMetaData()` et `onCuePoint()`, utilisez le code suivant :

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.onMetaData = metaDataHandler;
ns.onCuePoint = cuePointHandler;
ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
function cuePointHandler(infoObject:Object):void
{
    trace("cue point");
}
```

Définir la propriété « client » de l'objet NetStream comme this

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En donnant à la propriété `client` la valeur `this`, l'application recherche dans la portée des méthodes `onMetaData()` et `onCuePoint()`. Le code suivant en est un exemple :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Si les gestionnaires de rappel de `onMetaData` ou `onCuePoint` sont appelés et qu'il n'existe pas de méthode pour gérer le rappel, aucune erreur n'est générée. Pour gérer ces gestionnaires de rappel, créez des méthodes `onMetaData()` et `onCuePoint()` dans votre code, comme dans le fragment de code suivant :

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

Utilisation des points de repère et des métadonnées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes de rappel NetStream vous permettent de capturer et de traiter les événements de point de repère et de métadonnées lorsque la vidéo est en cours de lecture.

Utilisation des points de repère

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le tableau ci-dessous décrit les méthodes de rappel que vous pouvez utiliser pour capturer les points de repère F4V et FLV dans Flash Player et AIR.

Moteur d'exécution	**F4V**	**FLV**
Flash Player 9/ AIR1.0		OnCuePoint
		OnMetaData
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

L'exemple suivant utilise une boucle `for..in` simple pour effectuer une itération sur chaque propriété du paramètre `infoObject` que reçoit la fonction `onCuePoint()`. Il appelle à fonction trace() pour afficher un message lors de la réception de données de point de repère :

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function onCuePoint(infoObject:Object):void
{
    var key:String;
    for (key in infoObject)
    {
        trace(key + ": " + infoObject[key]);
    }
}
```

Le résultat suivant apparaît :

```
parameters:
name: point1
time: 0.418
type: navigation
```

Ce code utilise l'une des techniques disponibles pour définir l'objet pour lequel la méthode de rappel est exécutée. Vous pouvez utiliser d'autres techniques ; pour plus d'informations, voir « Ecriture de méthodes de rappel pour les métadonnées et les points de repère » à la page 502.

Utilisation des métadonnées de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser les fonctions `OnMetaData()` et `OnXMPData()` pour accéder à des informations sur les métadonnées dans votre fichier vidéo, y compris les points de repère.

Utilisation d'OnMetaData()

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Ces métadonnées comprennent des informations sur le fichier vidéo (durée, largeur et hauteur d'image, cadence). Les informations de métadonnées ajoutées à votre fichier vidéo dépendent du logiciel que vous utilisez pour coder le fichier vidéo.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);
ns.client = this;
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);

function onMetaData(infoObject:Object):void
{
    var key:String;
    for (key in infoObject)
    {
        trace(key + ": " + infoObject[key]);
    }
}
```

Le code précédent génère un résultat similaire à celui-ci :

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

Si la vidéo ne contient pas de son, les informations de métadonnées associées à l'audio (telles que `audiodatarate`) renvoient `undefined`, car aucune information audio n'est ajoutée aux métadonnées pendant l'encodage.

Dans le code ci-dessus, les informations de point de repère n'étaient pas affichées. Pour afficher les informations de point de repère, utilisez la fonction suivante, qui affiche de façon récursive les éléments d'un objet :

```
function traceObject(obj:Object, indent:uint = 0):void
{
    var indentString:String = "";
    var i:uint;
    var prop:String;
    var val:*;
    for (i = 0; i < indent; i++)
    {
        indentString += "\t";
    }
    for (prop in obj)
    {
        val = obj[prop];
        if (typeof(val) == "object")
        {
            trace(indentString + " " + prop + ": [Object]");
            traceObject(val, indent + 1);
        }
        else
        {
            trace(indentString + " " + prop + ": " + val);
        }
    }
}
```

L'utilisation du code ci-dessus pour suivre le paramètre `infoObject` de la méthode `onMetaData()` produit le résultat suivant :

```
width: 320
audiodatarate: 96
audiocodecid: 2
videocodecid: 4
videodatarate: 400
canSeekToEnd: true
duration: 16.334
audiodelay: 0.038
height: 213
framerate: 15
cuePoints: [Object]
    0: [Object]
        parameters: [Object]
            lights: beginning
        name: point1
        time: 0.418
        type: navigation
    1: [Object]
        parameters: [Object]
            lights: middle
        name: point2
        time: 7.748
        type: navigation
    2: [Object]
        parameters: [Object]
            lights: end
        name: point3
        time: 16.02
        type: navigation
```

L'exemple suivant affiche les métadonnées d'une vidéo MP4. Cet exemple suppose qu'il existe un objet TextArea appelé `metaDataOut` sur lequel il écrit les métadonnées.

```
package
{
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.events.NetStatusEvent;
    import flash.media.Video;
    import flash.display.StageDisplayState;
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.MouseEvent;

    public class onMetaDataExample extends Sprite
    {
        var video:Video = new Video();

        public function onMetaDataExample():void
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;

            addChild(video);
            video.x = 185;
            video.y = 5;

            video.attachNetStream(videoStream);

            videoStream.play("video.mp4");

            videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
        }

        public function onMetaData(infoObject:Object):void
        {
            for(var propName:String in infoObject)
            {
                metaDataOut.appendText(propName + "=" + infoObject[propName] + "\n");
            }
        }

        private function netStatusHandler(event:NetStatusEvent):void
        {
            if(event.info.code == "NetStream.Play.Stop")
                stage.displayState = StageDisplayState.NORMAL;
        }
    }
}
```

La fonction `onMetaData()` a produit le résultat suivant pour cette vidéo :

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

Utilisation de l'objet Information

Le tableau ci-dessous indique les valeurs possibles pour des métadonnées de vidéo qui sont transmises à la fonction de rappel `onMetaData()` dans l'objet qu'elles reçoivent :

Paramètre	Description
aacaot	Type d'objet audio AAC ; 0, 1 ou 2 pris en charge.
avclevel	Numéros de niveau AVC IDC, tels que 10, 11, 20, 21, et ainsi de suite.
avcprofile	Numéros de profil AVC, tel que 55, 77, 100, et ainsi de suite.
audiocodecid	Chaîne qui indique le codec audio (technique de codage/décodage) utilisé ; par exemple, « .Mp3 » ou « mp4a ».
audiodatarate	Nombre qui indique le taux d'encodage du son, en ko/s.
audiodelay	Nombre qui indique la valeur temporelle 0 du fichier FLV d'origine. Le contenu vidéo doit être légèrement retardé pour synchroniser correctement l'audio.
canSeekToEnd	Valeur booléenne définie sur `true` si le fichier FLV est codé avec une image-clé sur la dernière image, qui permet de rechercher jusqu'à la fin d'un fichier vidéo téléchargé progressivement. Elle est définie sur `false` si le fichier FLV n'est pas codé avec une image-clé sur la dernière image.
cuePoints	Tableau d'objets (un par point de repère intégré dans le fichier FLV). Cette valeur n'est pas définie si le fichier FLV ne contient pas de points de repère. Chaque objet possède les propriétés ci-dessous. • `type` : chaîne qui spécifie le type de point de repère : « navigation » ou « event ». • `name` : chaîne représentant le nom du point de repère. • `time` : nombre correspondant à l'heure du point de repère (en secondes) avec une précision de trois chiffres (millisecondes). • `parameters` : objet facultatif possédant des paires nom-valeur désignées par l'utilisateur au moment de la création des points de repère.
duration	Nombre indiquant la durée du fichier vidéo, en secondes.
framerate	Nombre indiquant la fréquence d'images du fichier FLV.
height	Nombre indiquant la hauteur du fichier FLV, en pixels.
seekpoints	Tableau qui répertorie les images-clés disponibles en tant que cachets d'horodatage, en millisecondes. Facultatif.
balises	Tableau de paires clé-valeur qui représente les informations dans l'atome « ilst » (l'équivalent des balises ID3 des fichiers MP4). iTunes utilise ces balises. Elles peuvent être utilisées pour afficher des illustrations, le cas échéant.
trackinfo	Objet qui fournit des informations sur toutes les pistes d'un fichier MP4, y compris sur l'ID de description de l'échantillonnage.
videocodecid	Chaîne indiquant la version codec utilisée pour coder la vidéo. Par exemple : « avc1 » ou « VP6F ».

Paramètre	Description
videodatarate	Nombre indiquant la vitesse de transmission vidéo du fichier FLV.
videoframerate	Cadence de la vidéo MP4.
width	Nombre indiquant la largeur du fichier FLV, en pixels.

Le tableau suivant répertorie les valeurs possibles du paramètre `videocodecid` :

videocodecid	Nom du codec
2	Sorenson H.263
3	Screen video (SWF versions 7 et les versions ultérieures uniquement)
4	VP6 (SWF versions 8 et les versions ultérieures uniquement)
5	VP6 avec canal alpha (SWF versions 8 et les versions ultérieures uniquement)

Le tableau suivant répertorie les valeurs possibles du paramètre `audiocodecid` :

audiocodecid	Nom du codec
0	non compressé
1	ADPCM
2	Mp3
4	Nellymoser @ 16 kHz mono
5	Nellymoser, 8kHz mono
6	Nellymoser
10	AAC
11	Speex

Utilisation d'onXMPData()

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La fonction de rappel `onXMPData()` reçoit des informations spécifiques à Adobe Extensible Metadata Platform (XMP) intégrée dans le fichier vidéo Adobe F4V ou FLV. Les métadonnées XMP contiennent des points de repère et d'autres métadonnées de vidéo. La prise en charge des métadonnées XMP a été intégrée à Flash Player 10 et Adobe AIR 1.5, et est assurée par les versions ultérieures.

L'exemple suivant traite les données de points de repère dans des métadonnées XMP :

```
package
{
    import flash.display.*;
    import flash.net.*;
    import flash.events.NetStatusEvent;
    import flash.media.Video;

    public class onXMPDataExample extends Sprite
    {
        public function onXMPDataExample():void
        {
            var videoConnection:NetConnection = new NetConnection();
            videoConnection.connect(null);

            var videoStream:NetStream = new NetStream(videoConnection);
            videoStream.client = this;
            var video:Video = new Video();

            addChild(video);

            video.attachNetStream(videoStream);

            videoStream.play("video.f4v");
        }

        public function onMetaData(info:Object):void {
            trace("onMetaData fired");
        }

        public function onXMPData(infoObject:Object):void
        {
            trace("onXMPData Fired\n");
            //trace("raw XMP =\n");
            //trace(infoObject.data);
            var cuePoints:Array = new Array();
            var cuePoint:Object;
            var strFrameRate:String;
            var nTracksFrameRate:Number;
            var strTracks:String = "";
            var onXMPXML = new XML(infoObject.data);
            // Set up namespaces to make referencing easier
            var xmpDM:Namespace = new Namespace("http://ns.adobe.com/xmp/1.0/DynamicMedia/");
            var rdf:Namespace = new Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#");
            for each (var it:XML in onXMPXML..xmpDM::Tracks)
            {
```

```
                var strTrackName:String =
it.rdf::Bag.rdf::li.rdf::Description.@xmpDM::trackName;
                var strFrameRateXML:String =
it.rdf::Bag.rdf::li.rdf::Description.@xmpDM::frameRate;
                strFrameRate = strFrameRateXML.substr(1,strFrameRateXML.length);

                nTracksFrameRate = Number(strFrameRate);

                strTracks += it;
            }
            var onXMPTracksXML:XML = new XML(strTracks);
            var strCuepoints:String = "";
            for each (var item:XML in onXMPTracksXML..xmpDM::markers)
            {
                strCuepoints += item;
            }
            trace(strCuepoints);
        }
    }
}
```

Pour un fichier vidéo court appelé startrekintro.f4v, cet exemple produit les lignes de suivi ci-dessous. Les lignes montrent les données des points de repère pour les points de repère de navigation et d'événement dans les métadonnées XMP :

```
onMetaData fired
onXMPData Fired

<xmpDM:markers xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/DynamicMedia/"
xmlns:stDim="http://ns.adobe.com/xap/1.0/sType/Dimensions#"
xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:x="adobe:ns:meta/">
  <rdf:Seq>
    <rdf:li>
      <rdf:Description xmpDM:startTime="7695905817600" xmpDM:name="Title1"
xmpDM:type="FLVCuePoint" xmpDM:cuePointType="Navigation">
        <xmpDM:cuePointParams>
          <rdf:Seq>
            <rdf:li xmpDM:key="Title" xmpDM:value="Star Trek"/>
            <rdf:li xmpDM:key="Color" xmpDM:value="Blue"/>
          </rdf:Seq>
        </xmpDM:cuePointParams>
      </rdf:Description>
    </rdf:li>
    <rdf:li>
      <rdf:Description xmpDM:startTime="10289459980800" xmpDM:name="Title2"
xmpDM:type="FLVCuePoint" xmpDM:cuePointType="Event">
        <xmpDM:cuePointParams>
          <rdf:Seq>
            <rdf:li xmpDM:key="William Shatner" xmpDM:value="First Star"/>
            <rdf:li xmpDM:key="Color" xmpDM:value="Light Blue"/>
          </rdf:Seq>
        </xmpDM:cuePointParams>
      </rdf:Description>
    </rdf:li>
  </rdf:Seq>
</xmpDM:markers>
onMetaData fired
```

Remarque : *dans les données XMP, le temps est stocké en unités DVA plutôt qu'en secondes Pour calculer le temps du point de repère, divisez l'heure de démarrage par la cadence. Par exemple, l'heure de démarrage de 7695905817600 divisé par une cadence de 254016000000 est égal à 30:30.*

Pour voir les métadonnées XMP brutes au complet, qui contiennent la cadence, retirez les identificateurs de commentaires (//) qui précèdent les deuxièeme et troisième instructions `trace()` au début de la fonction `onXMPData()`.

Pour plus d'informations sur XMP, voir :

- partners.adobe.com/public/developer/xmp/topic.html
- www.adobe.com/devnet/xmp/

Utilisation des métadonnées de l'image

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'événement `onImageData` envoie les données d'image sous la forme d'un tableau d'octets par l'intermédiaire d'un canal de données AMF0. Les données peuvent être au format JPEG, PNG ou GIF. Définissez une méthode de rappel `onImageData()` pour traiter ces informations, de la même manière que vous définiriez des méthodes de rappel pour `onCuePoint` et `onMetaData`. L'exemple suivant accède aux données d'image et les affiche à l'aide d'une méthode de rappel `onImageData()` :

```
public function onImageData(imageData:Object):void
{
    // display track number
    trace(imageData.trackid);
    var loader:Loader = new Loader();
    //imageData.data is a ByteArray object
    loader.loadBytes(imageData.data);
    addChild(loader);
}
```

Utilisation des métadonnées du texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'événement `onTextData` envoie des données de texte par le biais d'un canal de données AMF0. Les données de texte sont au format UTF-8 et contiennent des informations supplémentaires sur la mise en forme basées sur la spécification Timed Text 3GP. Cette spécification définit un format de sous-titrage normalisé. Définissez une méthode de rappel `onTextData()` pour traiter ces informations, de la même manière que vous définiriez des méthodes de rappel pour `onCuePoint` ou `onMetaData`. Dans l'exemple suivant, la méthode `onTextData()` affiche le numéro d'identification de la piste, ainsi que le texte correspondant à la piste.

```
public function onTextData(textData:Object):void
{
    // display the track number
    trace(textData.trackid);
    // displays the text, which can be a null string, indicating old text
    // that should be erased
    trace(textData.text);
}
```

Gestion de l'activité de l'objet NetStream

Flash Player 10.3 et les versions ultérieures, Adobe AIR 2.7 et les versions ultérieures

Vous pouvez gérer l'activité de l'objet NetStream de façon à recueillir les informations nécessaires à la prise en charge de l'analyse de l'utilisation de médias. Les fonctions de gestion décrites dans cette section permettent de créer des bibliothèques de mesure de médias contenant des informations non obligatoirement liées au lecteur vidéo affichant le média. Les développeurs de vos clients sont ainsi en mesure de choisir leurs lecteurs vidéo favoris lorsqu'ils utilisent votre bibliothèque. Utilisez la classe NetMonitor pour gérer la création d'objets NetStream et leur activité dans une application. La classe NetMonitor fournit une liste des objets NetStream actifs existant à un moment donné, et distribue un événement dès qu'un objet NetStream est créé.

Un objet NetStream distribue les événements répertoriés dans le tableau ci-dessous, en fonction du type de média en cours de lecture :

Evénement	Téléchargement progressif	Diffusion en continu RTMP	Diffusion en continu HTTP
NetStream.Play.Start	Oui	Oui	Non
NetStream.Play.Stop	Oui	Oui	Non
NetStream.Play.Complete	Oui	Oui	Non
NetStream.SeekStart.Notify	Oui	Oui	Oui
NetStream.Seek.Notify	Oui	Oui	Oui
NetStream.Unpause.Notify	Oui	Oui	Oui
NetStream.Unpause.Notify	Oui	Oui	Oui
NetStream.Play.Transition	Non applicable	Oui	Non applicable
NetStream.Play.TransitionComplete	Non applicable	Oui	Non applicable
NetStream.Buffer.Full	Oui	Oui	Oui
NetStream.Buffer.Flush	Oui	Oui	Oui
NetStream.Buffer.Empty	Oui	Oui	Oui

L'objet NetStreamInfo associé à l'occurrence de NetStream stocke également les dernières métadonnées et les derniers objets de données XMP trouvés dans le média.

Lors de la lecture du média via la diffusion en continu HTTP, les événements NetStream.Play.Start, NetStream.Play.Stop et NetStream.Play.Complete ne sont pas distribués, car l'application contrôle en intégralité le flux du média. Un lecteur vidéo doit synthétiser et distribuer ces événements pour les flux HTTP.

De même, les événements NetStream.Play.Transition et NetStream.Play.TransitionComplete ne sont pas distribués pour le téléchargement progressif ou le média HTTP. La commutation dynamique du taux d'échantillonnage est une fonction RTMP. Si un lecteur vidéo qui utilise le flux HTTP prend en charge une fonction similaire, il est alors en mesure de synthétiser et de distribuer des événements de transition.

Voir aussi

Adobe Developer Connection: Measuring video consumption in Flash (disponible en anglais uniquement)

Gestion des événements NetStream

Deux types d'événements fournissent des données d'utilisation utiles : `netStatus` et `mediaTypeData`. Il est par ailleurs possible d'utiliser un objet timer pour enregistrer régulièrement la position de la tête de lecture du NetStream.

Les événements `netStatus` fournissent des informations permettant de déterminer la quantité de données diffusées par un utilisateur. Les événements de tampon et de transition de flux RTMFP déclenchent également des événements `netStatus`.

Les événements `mediaTypeData` fournissent des métadonnées et des informations de données XMP. L'événement Netstream.Play.Complete est distribué en tant qu'événement `mediaTypeData`. D'autres données intégrées au flux sont également disponibles via les événements `mediaTypeData`, notamment les points de repère, le texte et les images.

L'exemple suivant explique comment créer une classe permettant de gérer les événements d'état et de données à partir de n'importe quel objet NetStream actif dans une application. En règle générale, une telle classe charge les données qu'elle souhaite analyser sur un serveur en vue de les stocker.

```
package com.adobe.example
{
    import flash.events.NetDataEvent;
    import flash.events.NetMonitorEvent;
    import flash.events.NetStatusEvent;
    import flash.net.NetMonitor;
    import flash.net.NetStream;

    public class NetStreamEventMonitor
    {
        private var netmon:NetMonitor;
        private var heartbeat:Timer = new Timer( 5000 );

        public function NetStreamEventMonitor()
        {
            //Create NetMonitor object
            netmon = new NetMonitor();
            netmon.addEventListener( NetMonitorEvent.NET_STREAM_CREATE, newNetStream );

            //Start the heartbeat timer
            heartbeat.addEventListener( TimerEvent.TIMER, onHeartbeat );
            heartbeat.start();
        }

        //On new NetStream
        private function newNetStream( event:NetMonitorEvent ):void
        {
            trace( "New Netstream object");
            var stream:NetStream = event.netStream;
            stream.addEventListener(NetDataEvent.MEDIA_TYPE_DATA, onStreamData);
            stream.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
        }

        //On data events from a NetStream object
        private function onStreamData( event:NetDataEvent ):void
        {

            var netStream:NetStream = event.target as NetStream;
            trace( "Data event from " + netStream.info.uri + " at " + event.timestamp );
            switch( event.info.handler )
            {
                case "onMetaData":
                    //handle metadata;
                    break;
                case "onXMPData":
                    //handle XMP;
                    break;
                case "onPlayStatus":
                    //handle NetStream.Play.Complete
                case "onImageData":
                    //handle image
                    break;
                case "onTextData":
```

```
                    //handle text
                    break;
                default:
                    //handle other events

            }
        }

        //On status events from a NetStream object
        private function onStatus( event:NetStatusEvent ):void
        {
            trace( "Status event from " + event.target.info.uri + " at " + event.target.time );
            //handle status events
        }
        //On heartbeat timer
        private function onHeartbeat( event:TimerEvent ):void
        {
            var streams:Vector.<NetStream> = netmon.listStreams();
            for( var i:int = 0; i < streams.length; i++ )
            {
                trace( "Heartbeat on " + streams[i].info.uri + " at " + streams[i].time );
                //handle heartbeat event
            }
        }

    }
}
```

Détection du domaine du lecteur

L'adresse URL et le domaine de la page Web sur laquelle un utilisateur visionne du contenu multimédia ne sont pas toujours immédiatement disponibles. Si le site Web hôte le permet, vous pouvez utiliser la classe ExternalInterface pour obtenir l'URL exacte. Néanmoins, certains sites Web prenant en charge des lecteurs vidéo tiers n'autorisent pas l'utilisation de la classe ExternalInterface. Dans ce cas, vous pouvez obtenir le domaine de la page Web actuelle grâce à la propriété `pageDomain` de la classe Security. L'adresse URL complète n'est pas divulguée afin d'assurer la sécurité et la confidentialité de l'utilisateur.

Le domaine de la page est disponible via la propriété `pageDomain` statique de la classe Security :

```
var domain:String = Security.pageDomain;
```

Rubriques avancées relatives aux fichiers vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les rubriques suivantes abordent certains problèmes d'utilisation des fichiers FLV.

A propos de la configuration de fichier FLV pour l'hébergement sur un serveur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour utiliser des fichiers FLV, il peut être nécessaire de configurer votre serveur pour le format de fichier FLV. Le protocole MIME (Multipurpose Internet Mail Extensions) est une spécification de données normalisée qui permet d'envoyer des fichiers non ASCII via des connexions Internet. Les navigateurs Web et les clients de courrier électronique sont configurés pour interpréter de nombreux types MIME afin qu'ils puissent envoyer et recevoir de la vidéo, de l'audio, des graphiques et du texte formaté. Pour charger des fichiers FLV à partir d'un serveur Web, il peut être nécessaire d'enregistrer l'extension de fichier et le type MIME auprès de votre serveur Web. Pour plus d'informations, consultez la documentation du serveur. Le type MIME des fichiers FLV est `video/x-flv`. Les informations complètes du type de fichier FLV se présentent comme suit :

- Type Mime : video/x-flv
- Extension du fichier : .flv
- Paramètres requis : aucun
- Paramètres facultatifs : aucun
- Considérations sur l'encodage : les fichiers FLV sont binaires, et une partie des applications peut nécessiter la définition du sous-type application/octet-stream
- Problèmes de sécurité : aucun
- Spécifications publiées : www.adobe.com/go/video_file_format_fr

Microsoft a changé la façon dont le multimédia en flux continu est géré par le serveur Web 6.0 IIS (Microsoft Internet Information Services) par rapport à ses versions antérieures. Les versions antérieures d'IIS ne nécessitent aucune modification pour diffuser en continu de la vidéo Flash. Dans IIS 6.0, le serveur Web fourni par défaut avec Windows 2003, le serveur nécessite un type MIME pour reconnaître que les fichiers FLV sont des flux continus multimédia.

Lorsque des fichiers SWF qui diffusent des fichiers FLV externes sont placés sur un serveur Microsoft Windows Server® 2003 et sont affichés dans un navigateur, le fichier SWF est lu correctement, mais la vidéo FLV n'est pas diffusée en continu. Ce problème a une incidence sur tous les fichiers FLV placés sur le serveur Windows Server 2003, y compris les fichiers créés avec des versions antérieures de l'outil de programmation Flash et le Kit Macromedia Flash Video Kit pour Dreamweaver MX 2004 d'Adobe. Ces fichiers fonctionnent correctement si vous les testez avec d'autres systèmes d'exploitation.

Pour plus d'informations sur la méthode de configuration de Microsoft Windows 2003 et Microsoft IIS Server 6.0 pour diffuser en continu de la vidéo FLV, voir la page www.adobe.com/go/tn_19439_fr.

Ciblage des fichiers FLV locaux sur Macintosh

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous tentez de lire un fichier FLV local à partir d'un lecteur non système sur un ordinateur Apple® Macintosh® avec un chemin qui utilise une barre oblique (/), il ne sera pas lu. Les lecteurs non système sont par exemple les lecteurs de CD-ROM, les disques durs partitionnés, les supports de stockage amovibles, les périphériques de stockage connectés, etc.

Remarque : *la raison de cet échec est une limitation du système d'exploitation, et non pas de Flash Player ni d'AIR.*

Pour qu'un fichier FLV puisse être lu à partir d'un lecteur non système sur un Macintosh, faites-y référence à l'aide d'un chemin absolu utilisant une notation à deux points (:) au lieu d'une notation à barres obliques (/). La liste suivante montre la différence entre les deux sortes de notation :

- Notation à barres obliques **:** myDrive/myFolder/myFLV.flv

- Notation à deux points **:** (Mac OS®) myDrive:myFolder:myFLV.flv

Exemple vidéo : Video Jukebox

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée un jukebox vidéo simple qui charge dynamiquement une liste de fichiers vidéos à lire en séquence. Vous pouvez ainsi créer une application qui permet à l'utilisateur de parcourir une série de didacticiels, ou encore qui permet de définir des publicités à afficher avant la vidéo demandée par l'utilisateur. Cet exemple illustre les fonctions suivantes d'ActionScript 3.0 :

- Actualisation de la position de la tête de lecture en fonction de la progression dans le fichier vidéo
- Détection et analyse des métadonnées d'un fichier vidéo
- Gestion de codes spécifiques dans un flux Internet
- Chargement, lecture, mise en pause et arrêt d'un fichier FLV chargé dynamiquement
- Redimensionnement d'un objet vidéo dans la liste d'affichage en fonction des métadonnées du flux reçu

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application Video Jukebox se trouvent dans le dossier Samples/VideoJukebox. L'application se compose des fichiers suivants :

Fichier	Description
VideoJukebox.fla ou VideoJukebox.mxml	Le fichier d'application principal pour Flex (MXML) ou Flash (FLA).
VideoJukebox.as	Classe comportant les principales fonctionnalités de l'application.
playlist.xml	Fichier comportant la liste des fichiers vidéo à charger dans le jukebox.

Chargement l'une liste de lecture vidéo externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le fichier externe playlist.xml contient la liste des vidéos à charger et leur ordre de lecture. Pour charger ce fichier XML, utilisez un objet URLLoader et un objet URLRequest, comme dans le code ci-dessous :

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

Ce code est placé dans le constructeur de la classe VideoJukebox, si bien que le fichier est chargé avant l'exécution de la suite du code. Dès que le chargement du fichier XML est terminé, la méthode `xmlCompleteHandler()` est appelée et analyse le fichier externe dans un objet XML, comme dans le code suivant :

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

L'objet XML `playlist` contient les données XML brutes du fichier externe, alors que videosXML est un objet XMLList qui ne contient que les nœuds vidéo. Le fragment de code suivant est un exemple de contenu du fichier playlist.xml :

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

Enfin, la méthode `xmlCompleteHandler()` appelle la méthode `main()` qui définit les diverses occurrences de composants dans la liste d'affichage, ainsi que les objets NetConnection et NetStream qui permettent de charger les fichiers FLV externes.

Création de l'interface utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour créer l'interface utilisateur, faites glisser cinq occurrences de l'objet Button sur la liste d'affichage et donnez-leur les noms d'occurrence suivants : `playButton`, `pauseButton`, `stopButton`, `backButton` et `forwardButton`.

Pour chacune de ces occurrences de Button, affectez un gestionnaire pour l'événement `click`, comme dans le fragment de code suivant :

```
playButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
pauseButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
stopButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
backButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
forwardButton.addEventListener(MouseEvent.CLICK, buttonClickHandler);
```

La méthode `buttonClickHandler()` utilise une instruction switch pour déterminer l'occurrence de bouton sur laquelle l'utilisateur a cliqué, comme dans le code suivant :

```
private function buttonClickHandler(event:MouseEvent):void
{
    switch (event.currentTarget)
    {
        case playButton:
            ns.resume();
            break;
        case pauseButton:
            ns.togglePause();
            break;
        case stopButton:
            ns.pause();
            ns.seek(0);
            break;
        case backButton:
            playPreviousVideo();
            break;
        case forwardButton:
            playNextVideo();
            break;
    }
}
```

Ajoutez ensuite une occurrence de Slider à la liste d'affichage, et donnez à cette occurrence le nom `volumeSlider`. Le code ci-dessous définit la propriété `liveDragging` de l'occurrence de Slider sur `true` et définit un écouteur d'événement pour l'événement `change` de cette occurrence :

```
volumeSlider.value = volumeTransform.volume;
volumeSlider.minimum = 0;
volumeSlider.maximum = 1;
volumeSlider.snapInterval = 0.1;
volumeSlider.tickInterval = volumeSlider.snapInterval;
volumeSlider.liveDragging = true;
volumeSlider.addEventListener(SliderEvent.CHANGE, volumeChangeHandler);
```

Ajoutez ensuite une occurrence de ProgressBar à la liste d'affichage, et donnez à cette occurrence le nom `positionBar`. Donnez à sa propriété `mode` la valeur « manual », comme ci-dessous :

```
positionBar.mode = ProgressBarMode.MANUAL;
```

Enfin, ajoutez une occurrence de l'objet Label à la liste d'affichage, et donnez à cette occurrence le nom `positionLabel`. La valeur de cette occurrence de l'objet Label sera définie par l'occurrence de timer.

Détection des métadonnées d'un objet vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque Flash Player détecte des métadonnées dans l'une des vidéos chargées, le gestionnaire de rappel `onMetaData()` est appelé pour la propriété `client` de l'objet NetStream. Le code suivant initialise un Object et configure le gestionnaire de rappel spécifié :

```
client = new Object();
client.onMetaData = metadataHandler;
```

La méthode `metadataHandler()` copie ses données dans la propriété meta définie par code au préalable. Vous pouvez ainsi accéder à tout moment aux métadonnées de la vidéo active, depuis n'importe quel point de l'application. L'objet Video sur la scène est ensuite redimensionné selon la taille renvoyées par les métadonnées. Enfin, l'occurrence de positionBar de la barre de progression est déplacée et redimensionnée en fonction de la taille de la vidéo en cours. Le code suivant est celui de la méthode `metadataHandler()` :

```
private function metadataHandler(metadataObj:Object):void
{
    meta = metadataObj;
    vid.width = meta.width;
    vid.height = meta.height;
    positionBar.move(vid.x, vid.y + vid.height);
    positionBar.width = vid.width;
}
```

Chargement dynamique d'une vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour charger dynamiquement chacune des vidéos, l'application utilise des objets NetConnection et NetStream. Le code suivant crée un objet NetConnection et passe la valeur `null` à la méthode `connect()`. Cette valeur `null` signifie que Flash Player va se connecter à une vidéo sur l'ordinateur local plutôt que sur un serveur tel que Flash Media Server.

Le code suivant crée les occurrences de NetConnection et NetStream, définit un écouteur d'événement pour l'événement `netStatus` et affecte l'objet `client` à la propriété `client` :

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

La méthode `netStatusHandler()` est appelée en cas de changement d'état de la vidéo. C'est le cas lorsque la lecture de la vidéo débute ou s'arrête, lorsque le signal est mis en mémoire tampon ou en cas d'impossibilité de trouver un flux vidéo. Le code suivant répertorie l'événement `netStatusHandler()` :

```
private function netStatusHandler(event:NetStatusEvent):void
{
    try
    {
        switch (event.info.code)
        {
            case "NetStream.Play.Start":
                t.start();
                break;
            case "NetStream.Play.StreamNotFound":
            case "NetStream.Play.Stop":
                t.stop();
                playNextVideo();
                break;
        }
    }
    catch (error:TypeError)
    {
        // Ignore any errors.
    }
}
```

Le code précédent évalue la propriété code de l'objet info et filtre les codes « NetStream.Play.Start », « NetStream.Play.StreamNotFound » et « NetStream.Play.Stop ». Les autres codes sont ignorés. Si le flux Internet débute, le code lance l'occurrence de Timer qui actualise la tête de lecture. Si le flux Internet est introuvable ou est interrompu, l'occurrence de Timer est arrêtée et l'application tente de lire la vidéo suivante dans la liste de lecture.

A chaque exécution de Timer, l'occurrence de la barre de progression `positionBar` actualise sa position en appelant la méthode `setProgress()` de la classe ProgressBar, et l'occurrence de Label `positionLabel` est actualisée avec le temps écoulé et la durée totale de la vidéo active.

```
private function timerHandler(event:TimerEvent):void
{
    try
    {
        positionBar.setProgress(ns.time, meta.duration);
        positionLabel.text = ns.time.toFixed(1) + " of " meta.duration.toFixed(1) + " seconds";
    }
    catch (error:Error)
    {
        // Ignore this error.
    }
}
```

Contrôle du volume de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez contrôler le volume audio de la vidéo chargée dynamiquement par le biais de la propriété `soundTransform` de l'objet NetStream. L'application Video jukebox permet de modifier le volume en changeant la valeur de l'occurrence de Slider `volumeSlider`. Le code suivant montre comment changer le volume en affectant la valeur du composant Slider à un objet SoundTransform qui est lui-même affecté à la propriété `soundTransform` de l'objet NetStream :

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

Contrôle de la lecture de la vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le reste de l'application contrôle la lecture de la vidéo lorsque la fin du flux vidéo est atteinte ou lorsque l'utilisateur change de vidéo.

La méthode suivante obtient l'adresse URL de la vidéo à partir de l'objet XMLList pour l'index sélectionné :

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

La méthode `playVideo()` appelle la méthode `play()` de l'objet NetStream pour charger la vidéo sélectionnée :

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

La méthode `playPreviousVideo()` décrémente l'index vidéo sélectionné, appelle la méthode `playVideo()` pour charger le nouveau fichier vidéo et rend la barre de progression visible :

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

La méthode finale, `playNextVideo()`, incrémente l'index vidéo et rappelle la méthode `playVideo()`. Si la vidéo en cours est la dernière de la liste de lecture, la méthode `clear()` est appelée pour l'objet Video et la propriété `visible` de l'occurrence de la barre de progression est mise à `false` :

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

Présentation à accélération matérielle par le biais de la classe StageVideo

Flash Player 10.2 et les versions ultérieures

Flash Player améliore les performances vidéo par le biais d'une accélération matérielle pour le décodage H.264. Vous pouvez optimiser ces performances à l'aide de l'API StageVideo. La vidéo sur la scène permet à l'application d'exploiter la présentation à accélération matérielle.

Les moteurs d'exécution suivants prennent en charge l'API StageVideo :

- Flash Player 10.2 et les versions ultérieures

Remarque : *dans Flash Player 11.4/AIR 3.4 et les versions ultérieures, vous pouvez utiliser l'entrée caméra avec StageVideo.*


Vous trouverez le code source à télécharger et d'autres informations relatives à la fonction de vidéo sur la scène à l'adresse Getting Started with Stage Video (disponible en anglais uniquement).


Pour consulter un didacticiel de démarrage rapide, voir Working with Stage Video (disponible en anglais uniquement).

Présentation de l'accélération matérielle basée sur la classe StageVideo

La présentation à accélération matérielle, qui comprend la mise à l'échelle de la vidéo, la conversion de la couleur et la fusion, exploite l'optimisation des performances assurée par le décodage à accélération matérielle. Sur les périphériques qui gèrent l'accélération par processeur graphique (matérielle), vous disposez d'un objet flash.media.StageVideo pour traiter la vidéo directement sur le matériel du périphérique. Un traitement direct permet à l'unité centrale d'exécuter d'autres tâches pendant que le processeur graphique gère la vidéo. La classe Video existante fait en revanche appel à la présentation logicielle. La présentation logicielle est exécutée dans l'unité centrale et sollicite parfois une proportion élevée des ressources du système.

A l'heure actuelle, peu de périphériques prennent en charge l'accélération par processeur graphique complète. La vidéo sur la scène permet toutefois aux applications d'exploiter pleinement l'accélération matérielle éventuellement intégrée.

La classe StageVideo ne rend pas obsolète la classe Video. L'association de ces deux classes assure l'expérience optimale de visualisation de la vidéo autorisée par les ressources du périphérique à un moment donné. L'application exploite l'accélération matérielle en écoutant les événements appropriés et bascule entre les classes StageVideo et Video en fonction des besoins.

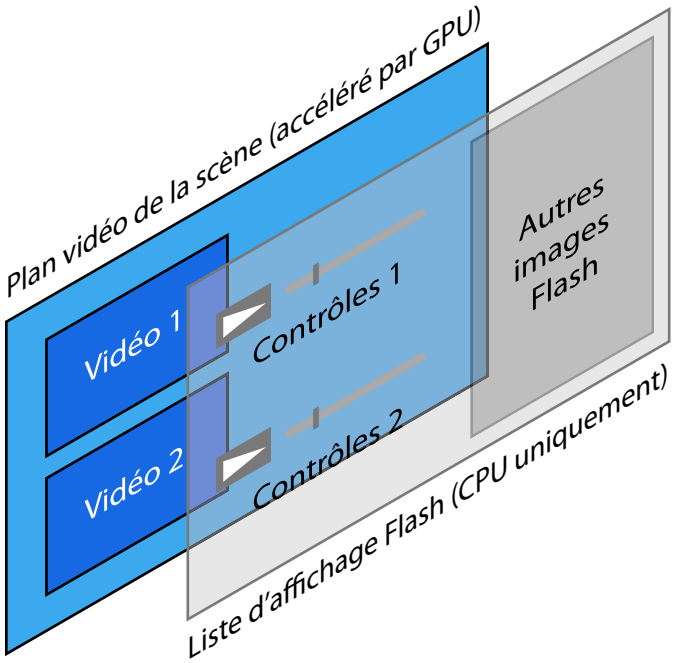
La classe StageVideo impose diverses restrictions en matière d'utilisation de la vidéo. Avant de mettre en œuvre la classe StageVideo, passez en revue les directives et assurez-vous que l'application est en mesure de les prendre en charge. Si vous acceptez les restrictions, utilisez la classe StageVideo à chaque fois que Flash Player détecte qu'une présentation à accélération matérielle est disponible. Voir « Directives et restrictions » à la page 530.

Plans parallèles : Vidéo sur la scène et liste d'affichage Flash

Le modèle de vidéo sur la scène permet à Flash Player de séparer la vidéo de la liste d'affichage. Flash Player divise l'affichage composite entre deux plans Z :

Plan de vidéo sur la scène Le plan de vidéo sur la scène est situé en arrière-plan. Il n'affiche que la vidéo à accélération matérielle. Ce plan n'est de ce fait disponible que si l'accélération matérielle n'est pas prise en charge par le périphérique ou qu'elle n'est pas intégrée à ce dernier. En ActionScript, les objets StageVideo traitent les vidéos lues sur le plan de la vidéo sur la scène.

Plan associé à la liste d'affichage de Flash Les entités qui figurent dans la liste d'affichage de Flash sont combinées sur un plan situé devant le plan de la vidéo sur la scène. Ces entités incluent tout élément rendu par le moteur d'exécution, y compris les contrôles de lecture. Si l'accélération matérielle n'est pas disponible, les vidéos sont lues sur ce plan uniquement, à l'aide de l'objet de la classe Video. La vidéo sur la scène est toujours affichée derrière les graphiques de la liste d'affichage Flash.



Plans d'affichage de la vidéo

L'objet StageVideo apparaît dans une zone rectangulaire sans rotation de l'écran, alignée sur la fenêtre. Il est impossible de superposer les objets derrière le plan de la vidéo sur la scène. Le plan associé à la liste d'affichage de Flash permet toutefois de superposer d'autres graphiques au plan de la vidéo sur la scène. La vidéo sur la scène et la liste d'affichage s'exécutent simultanément. Vous pouvez de ce fait combiner les deux mécanismes pour créer un effet visuel unifié qui utilise deux plans discrets. Vous pouvez, par exemple, réserver le plan avant aux contrôles de lecture associés à la vidéo sur la scène qui s'exécute en arrière-plan.

Vidéo sur la scène et codec H.264

Dans les applications Flash Player, la mise en œuvre de l'accélération matérielle de la vidéo se compose de deux étapes :

1 Encodage de la vidéo au format H.264

2 Mise en œuvre de l'API StageVideo

Pour obtenir les meilleurs résultats possibles, exécutez les deux étapes. Le codec H.264 permet d'exploiter pleinement l'accélération matérielle, de la phase de décodage de la vidéo à la phase de présentation.

La vidéo sur la scène élimine la rétro-lecture du processeur graphique à l'unité centrale. En d'autres termes, le processeur graphique n'envoie plus d'images décodées à l'unité centrale en vue de leur composition avec les objets de la liste d'affichage. Il fusionne plutôt directement à l'écran les images décodées et rendues, derrière les objets issus de la liste d'affichage. Cette technique sollicite moins l'unité centrale et la mémoire tout en assurant une fidélité accrue des pixels.

Voir aussi

« Présentation des formats vidéo » à la page 490

Directives et restrictions

Si la vidéo est exécutée en mode Plein écran, la vidéo sur la scène est toujours disponible, sous réserve que le périphérique prenne en charge l'accélération matérielle. Flash Player s'exécute toutefois aussi au sein d'un navigateur. Dans le contexte du navigateur, le paramètre `wmode` affecte la disponibilité de la vidéo sur la scène. Tentez d'utiliser systématiquement le paramètre `wmode="direct"` si vous souhaitez faire appel à la vidéo sur la scène. La vidéo sur la scène est incompatible avec d'autres paramètres `wmode` si le mode Plein écran est désactivé. Cette restriction signifie qu'à l'exécution, la disponibilité continue de la vidéo sur la scène n'est pas assurée. Si, par exemple, l'utilisateur quitte le mode Plein écran alors que la vidéo sur la scène est en cours de lecture, le contexte de cette dernière correspond à nouveau au navigateur. Si le paramètre `wmode` du navigateur n'est pas défini sur `« direct »`, la vidéo sur la scène risque de n'être soudainement plus disponible. Flash Player communique les changements de contexte de lecture aux applications par le biais d'une série d'événements. Si vous mettez en œuvre l'API StageVideo, conservez un objet Video à titre d'objet de secours au cas où la vidéo de la scène ne serait plus disponible.

En raison de son lien direct avec le matériel, la vidéo sur la scène interdit certaines fonctionnalités vidéo. La vidéo sur la scène impose les contraintes suivantes :

- Pour chaque fichier SWF, Flash Player limite à quatre le nombre d'objets StageVideo capable de visionner simultanément des vidéos. Selon les ressources matérielles dont dispose le périphérique, la limite réelle risque toutefois d'être encore inférieure.
- La vidéo n'est pas synchronisée avec le contenu qu'affiche le moteur d'exécution.
- La zone d'affichage vidéo ne peut être qu'un rectangle. Il est impossible d'utiliser des zones d'affichage plus avancées, telles que des formes elliptiques ou irrégulières.
- Il est impossible de faire pivoter la vidéo.
- Il est impossible de placer la vidéo dans le cache de bitmaps ou d'utiliser BitmapData pour y accéder.

- Il est impossible d'appliquer des filtres à la vidéo.
- Il est impossible d'appliquer des transformations de couleur à la vidéo.
- Il est impossible d'appliquer une valeur alpha à la vidéo.
- Les modes de fusion que vous appliquez aux objets de la liste d'affichage ne s'appliquent pas à la vidéo sur la scène.
- Vous pouvez positionner la vidéo uniquement sur les limites de pixels pleines.
- Bien que le rendu par processeur graphique soit optimisé en fonction du matériel du périphérique indiqué, il n'est pas identique « au pixel près » d'un périphérique à un autre. De légères variations risquent de survenir en raison des différences de pilotes et de plates-formes.
- Quelques périphériques ne prennent pas en charge tous les espaces colorimétriques requis. Par exemple, certains périphériques ne prennent pas en charge BT.709, le standard H.264. Dans ce cas de figure, vous pouvez utiliser BT.601 pour assurer un affichage rapide.
- Vous ne pouvez pas utiliser la vidéo sur la scène avec des paramètres WMODE tels que normal, opaque ou transparent. La vidéo sur la scène ne prend en charge `WMODE=direct` que si le mode Plein écran est désactivé. WMODE n'a aucun effet dans Safari 4 ou version ultérieure, et IE 9 ou une version ultérieure.

Dans la plupart des cas, ces restrictions n'affectent pas les lecteurs vidéo. Si vous êtes prêt à les accepter, utilisez dans la mesure du possible la vidéo sur la scène.

Voir aussi

« Utilisation du mode Plein écran » à la page 173

Utilisation des API StageVideo

La vidéo sur la scène est un mécanisme intégré au moteur d'exécution qui optimise la lecture de vidéos et les performances des périphériques. Le moteur d'exécution crée et gère ce mécanisme. En tant que développeur, il vous incombe de configurer l'application de sorte à l'exploiter pleinement.

Pour utiliser la vidéo sur la scène, vous mettez en œuvre une structure de gestionnaires d'événement qui détectent la disponibilité ou la non-disponibilité de la vidéo sur la scène. Lorsque vous êtes averti de la disponibilité de la vidéo sur la scène, vous récupérez un objet StageVideo de la propriété `Stage.stageVideos`. Le moteur d'exécution insère dans l'objet Vector un ou plusieurs objets StageVideo. Vous pouvez alors afficher une vidéo en flux continu par le biais de l'un des objets StageVideo fournis, plutôt qu'un objet Video.

Sur Flash Player, lorsque vous êtes averti que la vidéo sur la scène n'est plus disponible, réactivez l'utilisation d'un objet Video pour afficher le flux vidéo.

Remarque : *il est impossible de créer des objets StageVideo.*

Propriété Stage.stageVideos

La propriété `Stage.stageVideos` est un objet Vector qui permet d'accéder aux occurrences de StageVideo. Selon les ressources matérielles et système disponibles, ce vecteur peut contenir jusqu'à quatre objets StageVideo. Il est possible de limiter à un ou zéro les périphériques mobiles.

Si la vidéo sur la scène n'est pas disponible, ce vecteur ne contient pas d'objet. Pour éviter les erreurs d'exécution, veillez à n'accéder aux membres de cet objet vectoriel que si l'événement `StageVideoAvailability` le plus récent indique la disponibilité de la vidéo sur la scène.

Evénements StageVideo

La structure de l'API StageVideo propose les événements suivants :

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY Evénement envoyé en cas de modification de la propriété `Stage.stageVideos`. La propriété `StageVideoAvailabilityEvent.availability` indique soit `AVAILABLE`, soit `UNAVAILABLE`. Cet événement permet de déterminer si la propriété `stageVideos` contient des objets StageVideo, plutôt que de vérifier directement la longueur de l'objet vectoriel `Stage.stageVideos`.

StageVideoEvent.RENDER_STATE Envoyé lorsqu'un objet NetStream ou Camera a été associé à un objet StageVideo et est en cours d'exécution. Indique le type de décodage actuellement utilisé : matériel, logiciel ou non disponible (aucune valeur affichée). L'événement cible contient les propriétés `videoWidth` et `videoHeight`, dont l'utilisation est adaptée au redimensionnement de la fenêtre d'affichage de la vidéo.

Important : *étant donné qu'elles ne figurent pas dans la liste d'affichage standard, les coordonnées issues de l'objet StageVideo cible utilisent les coordonnées Stage.*

VideoEvent.RENDER_STATE Evénement envoyé si un objet Video est en cours d'utilisation. Il indique le type de décodage en cours d'utilisation (matériel ou logiciel). Si cet événement indique un décodage à accélération matérielle, utilisez de préférence un objet StageVideo. L'événement Video cible contient les propriétés `videoWidth` et `videoHeight`, dont l'utilisation est adaptée au redimensionnement de la fenêtre d'affichage de la vidéo.

Flux de travail de mise en œuvre de la fonction StageVideo

Procédez comme suit pour mettre en œuvre la fonction StageVideo :

1. Ecoutez l'événement `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` pour savoir quand l'objet vectoriel `Stage.stageVideos` a changé. Voir « Utilisation de l'événement StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY » à la page 533.
2. Si l'événement `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` indique que la vidéo sur la scène est disponible, utilisez l'objet vectoriel `Stage.stageVideos` à l'intérieur du gestionnaire d'événement pour accéder à un objet StageVideo.
3. Associez un objet NetStream à l'aide de `StageVideo.attachNetStream()` ou associez un objet Camera à l'aide de `StageVideo.attachCamera()`.
4. Lisez la vidéo à l'aide de `NetStream.play()`.
5. Ecoutez l'événement `StageVideoEvent.RENDER_STATE` sur l'objet StageVideo afin de déterminer l'état de lecture de la vidéo. La réception de cet événement indique également que les propriétés width et height de la vidéo ont été initialisées ou modifiées. Voir « Utilisation des événements StageVideoEvent.RENDER_STATE et VideoEvent.RENDER_STATE » à la page 535.
6. Ecoutez l'événement `VideoEvent.RENDER_STATE` sur l'objet Video. Cet événement indique les mêmes états que l'événement `StageVideoEvent.RENDER_STATE` ; vous pouvez donc également l'utiliser pour déterminer si l'accélération par processeur graphique est disponible. La réception de cet événement indique également que les propriétés width et height de la vidéo ont été initialisées ou modifiées. Voir « Utilisation des événements StageVideoEvent.RENDER_STATE et VideoEvent.RENDER_STATE » à la page 535.

Initialisation des écouteurs d'événements StageVideo

Configurez les écouteurs StageVideoAvailabilityEvent et VideoEvent lors de l'initialisation de l'application. Vous pouvez, par exemple, initialiser ces écouteurs dans le gestionnaire d'événement `flash.events.Event.ADDED_TO_STAGE`. Cet événement assure la visibilité de l'application sur la scène :

```
public class SimpleStageVideo extends Sprite
    private var nc:NetConnection;
    private var ns:NetStream;

    public function SimpleStageVideo()
    {
        // Constructor for SimpleStageVideo class
        // Make sure the app is visible and stage available
        addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
    }

    private function onAddedToStage(event:Event):void
    {
        //...
        // Connections
        nc = new NetConnection();
        nc.connect(null);
        ns = new NetStream(nc);
        ns.addEventListener(NetStatusEvent.NET_STATUS, onNetStatus);
        ns.client = this;

        // Screen
        video = new Video();
        video.smoothing = true;

        // Video Events
        // the StageVideoEvent.STAGE_VIDEO_STATE informs you whether
        // StageVideo is available
        stage.addEventListener(StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY,
            onStageVideoState);
        // in case of fallback to Video, listen to the VideoEvent.RENDER_STATE
        // event to handle resize properly and know about the acceleration mode running
        video.addEventListener(VideoEvent.RENDER_STATE, videoStateChange);
        //...
    }
```

Utilisation de l'événement StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY

Dans le gestionnaire `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`, optez pour un objet Video ou StageVideo selon la disponibilité de StageVideo. Si la propriété `StageVideoAvailabilityEvent.availability` est définie sur `StageVideoAvailability.AVAILABLE`, utilisez StageVideo. Dans ce cas de figure, attendez-vous à ce que le vecteur Stage.stageVideos contienne un ou plusieurs objets StageVideo. Obtenez un objet StageVideo à partir de la propriété `Stage.stageVideos` et associez-lui l'objet NetStream. Etant donné que les objets StageVideo s'affichent toujours en arrière-plan, supprimez tout objet Video existant (toujours affiché au premier plan). Ce gestionnaire d'événement permet également d'associer un écouteur à l'événement `StageVideoEvent.RENDER_STATE`.

Si la propriété `StageVideoAvailabilityEvent.availability` est définie sur `StageVideoAvailability.UNAVAILABLE`, n'utilisez pas l'objet StageVideo et n'accédez pas à l'objet vectoriel `Stage.stageVideos`. Associez dans ce cas l'objet NetStream à un objet Video. Ajoutez enfin l'objet StageVideo ou Video à la scène et appelez la méthode `NetStream.play()`.

Le code suivant illustre le traitement de l'événement `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` :

```
private var sv:StageVideo;
private var video:Video;

private function onStageVideoState(event:StageVideoAvailabilityEvent):void
{
    // Detect if StageVideo is available and decide what to do in toggleStageVideo
    toggleStageVideo(event.availability == StageVideoAvailability.AVAILABLE);
}

private function toggleStageVideo(on:Boolean):void
{
    // To choose StageVideo attach the NetStream to StageVideo
    if (on)
    {
        stageVideoInUse = true;
        if ( sv == null )
        {
            sv = stage.stageVideos[0];
            sv.addEventListener(StageVideoEvent.RENDER_STATE, stageVideoStateChange);
                sv.attachNetStream(ns);
        }

        if (classicVideoInUse)
        {
            // If you use StageVideo, remove from the display list the
            // Video object to avoid covering the StageVideo object
            // (which is always in the background)
            stage.removeChild ( video );
            classicVideoInUse = false;
        }
    } else
    {
        // Otherwise attach it to a Video object
        if (stageVideoInUse)
            stageVideoInUse = false;
        classicVideoInUse = true;
        video.attachNetStream(ns);
        stage.addChildAt(video, 0);
    }

    if ( !played )
    {
        played = true;
        ns.play(FILE_NAME);
    }
}
```

Important : *la première fois qu'une application accède à l'élément vectoriel au niveau Stage.stageVideos[0], le rectangle par défaut est défini sur 0,0,0,0 et les propriétés de panoramique et zoom utilisent les valeurs par défaut. Réinitialisez toujours ces valeurs sur les paramètres de votre choix. Vous disposez des propriétés* `videoWidth` *et* `videoHeight` *de l'événement cible* `StageVideoEvent.RENDER_STATE` *ou* `VideoEvent.RENDER_STATE` *pour calculer les dimensions de la fenêtre d'affichage de la vidéo.*

Téléchargez le code source complet de cet exemple d'application à l'adresse Getting Started with Stage Video (disponible en anglais uniquement).

Utilisation des événements StageVideoEvent.RENDER_STATE et VideoEvent.RENDER_STATE

Les objets StageVideo et Video envoient des événements qui indiquent aux applications tout changement de l'environnement d'affichage. Ces événements sont `StageVideoEvent.RENDER_STATE` et `VideoEvent.RENDER_STATE`.

Un objet StageVideo ou Video distribue un événement d'état de rendu lorsqu'un objet NetStream est associé et que sa lecture débute. Il envoie également cet événement lorsque l'environnement d'affichage est modifié (en cas de redimensionnement de la fenêtre d'affichage de la vidéo, par exemple). Ces notifications permettent de réinitialiser la fenêtre d'affichage sur les valeurs `videoHeight` et `videoWidth` actuelles de l'objet événement cible.

Les états de rendu indiqués sont les suivants :

- `RENDER_STATUS_UNAVAILABLE`
- `RENDER_STATUS_SOFTWARE`
- `RENDER_STATUS_ACCELERATED`

Les états de rendu indiquent quand le décodage à accélération matérielle est en cours d'utilisation, quelle que soit la classe qui lit actuellement la vidéo. Vérifiez la propriété `StageVideoEvent.status` pour savoir si le décodage requis est disponible. Si elle est définie sur « unavailable », l'objet StageVideo ne peut pas lire la vidéo. Cet état nécessite que vous associiez à nouveau immédiatement l'objet NetStream à un objet Video. Les autres états indiquent à l'application les conditions de rendu en cours.

Le tableau ci-dessous décrit les implications de toutes les valeurs d'état de rendu pour les objets StageVideoEvent et VideoEvent dans Flash Player :

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	Le décodage et la présentation relèvent tous deux du matériel (performances optimales).	La présentation relève du matériel, le décodage du logiciel (performances satisfaisantes).	Aucune ressource du processeur graphique n'est disponible pour traiter la vidéo et aucun élément n'est affiché. **Utilisez un objet vidéo.**
VideoEvent	La présentation relève du logiciel, le décodage du matériel. (Performances satisfaisantes sur un système de bureau moderne uniquement. Elles sont dégradées en mode Plein écran.)	La présentation et le décodage relèvent tous deux du logiciel. (Performances les moins satisfaisantes. Elles sont dégradées en mode Plein écran.)	(S/O)

Espaces colorimétriques

La vidéo sur la scène fait appel aux capacités matérielles sous-jacentes pour prendre en charge les espaces colorimétriques. Un contenu SWF peut fournir des métadonnées qui indiquent l'espace colorimétrique préféré. Le matériel graphique du périphérique détermine toutefois s'il est possible d'utiliser l'espace colorimétrique. Un périphérique peut prendre en charge plusieurs espaces colorimétriques, tandis qu'un autre périphérique n'en prend en charge aucun. Si le matériel ne prend pas en charge l'espace colorimétrique requis, Flash Player tente d'identifier la correspondance la plus proche parmi les espaces colorimétriques gérés.

Pour déterminer les espaces colorimétriques pris en charge par le matériel, faites appel à la propriété `StageVideo.colorSpaces`. Cette propriété renvoie la liste des espaces colorimétriques pris en charge dans un vecteur String :

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

Pour savoir quel espace colorimétrique utilise la vidéo en cours de lecture, faites appel à la propriété `StageVideoEvent.colorSpace`. Vérifiez cette propriété dans le gestionnaire associé à l'événement `StageVideoEvent.RENDER_STATE` :

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Si Flash Player ne détecte pas de valeur de substitution pour un espace colorimétrique non pris en charge, la vidéo sur la scène utilise l'espace colorimétrique par défaut, BT.601. Ainsi, les flux vidéo à codage H.264 utilisent généralement l'espace colorimétrique BT.709. Si le matériel du périphérique ne prend pas en charge BT.709, la propriété `colorSpace` renvoie `« BT601 »`. La valeur `StageVideoEvent.colorSpace« unknown »` indique que le matériel ne fournit pas de technique d'identification de l'espace colorimétrique.

Si l'application considère que l'espace colorimétrique actif est inacceptable, vous pouvez basculer d'un objet StageVideo à un objet Video. La classe Video prend en charge tous les espaces colorimétriques par le biais d'une composition logicielle.

Chapitre 26 : Utilisation de caméras

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La source des données vidéo peut être une caméra connectée à l'ordinateur de l'utilisateur, et il est possible de gérer en ActionScript l'affichage et la manipulation de ces données. La classe Camera est le mécanisme intégré à ActionScript permettant d'utiliser un ordinateur ou une caméra.

Sur les périphériques mobiles, vous pouvez également utiliser la classe CameraUI. La classe CameraUI ouvre une application de caméra indépendante pour permettre à l'utilisateur de capturer un cliché ou d'enregistrer une vidéo. Lorsque l'utilisateur a terminé, votre application peut accéder à l'image ou à la vidéo via un objet MediaPromise.

Voir aussi

Christian Cantrell : How to use CameraUI in a Cross-platform Way (disponible en anglais uniquement)

Michaël CHAIZE : Android, AIR and the Camera (disponible en anglais uniquement)

Présentation de la classe Camera

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'objet Camera permet d'établir une connexion avec la caméra locale de l'utilisateur et de diffuser le signal vidéo soit localement (à destination de l'utilisateur lui-même) ou à destination d'un serveur tel que Flash Media Server.

La classe Camera permet d'accéder aux informations suivantes sur la caméra de l'utilisateur :

- Quelles caméras installées sur l'ordinateur ou le périphérique de l'utilisateur sont-elles disponibles ?
- Installation d'une caméra ou non
- Si Flash Player est autorisé ou non à accéder à la caméra de l'utilisateur
- Caméra active
- Largeur et hauteur de la vidéo en cours de capture, en pixels

La classe Camera fournit les méthodes et les propriétés qui permettent d'utiliser les objets Camera. Par exemple, la propriété statique `Camera.names` contient le tableau des noms des caméras actuellement installées sur l'ordinateur. Par ailleurs, la propriété `name` permet d'afficher le nom de la caméra active.

Remarque : *si vous diffusez en continu la vidéo émanant d'une caméra via le réseau, vous devez toujours traiter les interruptions du réseau. Ces interruptions se produisent pour diverses raisons, en particulier sur les périphériques mobiles.*

Affichage du contenu de la caméra

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La connexion à une caméra peut nécessiter moins de code que l'utilisation des classes NetConnection et NetStream pour charger un fichier vidéo. Par contre, l'utilisation de la classe Camera peut parfois s'avérer délicate, car avec Flash Player, il est nécessaire d'avoir l'autorisation de l'utilisateur pour se connecter à sa caméra et la rendre accessible par code.

Le code suivant montre comment utiliser la classe Camera pour établir une connexion avec la caméra de l'utilisateur :

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Remarque : *la classe Camera ne possède pas de méthode constructeur. Pour créer une nouvelle occurrence de Camera, utilisez la méthode statique* `Camera.getCamera()`.

Conception d'une application gérant une caméra locale

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lors de la programmation d'une application destinée à gérer la caméra de l'utilisateur, prenez les précautions suivantes :

- Vérifiez qu'une caméra est installée sur l'ordinateur de l'utilisateur. Gérez le cas de figure dans lequel aucune caméra n'est disponible.
- Pour Flash Player uniquement, vérifiez si l'utilisateur a autorisé l'accès à la caméra de façon explicite. Pour des raisons de sécurité, le lecteur Flash affiche la boîte de dialogue de paramétrage de Flash Player, qui permet à l'utilisateur d'autoriser ou non l'accès à sa caméra. Flash Player ne peut donc pas établir une connexion avec la caméra d'un utilisateur et diffuser un signal vidéo sans la permission de cet utilisateur. Si l'utilisateur clique sur le bouton d'autorisation, votre application peut se connecter à sa caméra. Si l'utilisateur clique sur le bouton de refus, votre application ne pourra pas établir de liaison avec sa caméra. Dans les deux cas, votre application doit gérer élégamment la réponse.
- Pour AIR uniquement, vérifiez que la classe Camera est prise en charge par les profils de périphériques gérés par l'application.
- Elle n'est pas prise en charge par les navigateurs mobiles.
- Elle n'est pas prise en charge par les applications AIR mobiles qui font appel au mode de rendu par processeur graphique.
- Sur les périphériques mobiles, il n'est possible d'activer qu'une seule caméra à la fois.

Voir aussi

Christophe Coenraets : Multi-User Video Tic-Tac-Toe (disponible en anglais uniquement)

Mark Doherty : Android Radar app (source)

Etablissement d'une connexion avec la caméra de l'utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour établir une connexion avec la caméra de l'utilisateur, la première étape consiste à créer une occurrence de l'objet Camera, en créant une variable du type Camera et en l'initialisant avec la valeur renvoyée par la méthode statique `Camera.getCamera()`.

L'étape suivante consiste à créer un objet Video et à lui affecter l'objet Camera.

La troisième étape consiste à ajouter cet objet Video à la liste d'affichage. Les étapes 2 et 3 sont nécessaires, car la classe Camera n'étend pas la classe DisplayObject, il est donc impossible de l'ajouter directement à la liste d'affichage. Pour afficher le signal vidéo provenant de la caméra, vous devez ensuite créer un autre objet Video et appeler la méthode `attachCamera()`.

Le code suivant illustre ces trois opérations :

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Notez que si aucune caméra n'est installée sur l'ordinateur de l'utilisateur, l'application n'affiche rien.

Pour une application réelle, d'autres tâches sont nécessaires. Pour plus d'informations, voir « Vérification de la présence de caméras » à la page 539 et « Détection de l'autorisation d'accéder à la caméra » à la page 540.

Voir aussi

Lee Brimelow : How to access the camera on Android devices (disponible en anglais uniquement)

Vérification de la présence de caméras

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Avant de tenter d'utiliser les méthodes ou propriétés d'une occurrence de Camera, il est préférable de vérifier qu'une caméra est bien installée. Il existe deux techniques pour vérifier la présence d'une ou plusieurs caméras :

- Tester la propriété statique `Camera.names`, qui contient le tableau des noms des caméras disponibles. Ce tableau ne comporte en général qu'une seule chaîne au maximum, car la plupart des utilisateurs ne disposent au mieux que d'une seule caméra à la fois. Le code suivant montre comment tester la propriété `Camera.names` pour vérifier que l'utilisateur dispose d'au moins une caméra :

```
if (Camera.names.length > 0)
{
    trace("User has at least one camera installed.");
    var cam:Camera = Camera.getCamera(); // Get default camera.
}
else
{
    trace("User has no cameras installed.");
}
```

- Vérifier la valeur renvoyée de la méthode statique `Camera.getCamera()`. Si aucune caméra n'est disponible ou installée, la méthode renvoie `null`, sinon elle renvoie une référence à un objet Camera. Le code suivant montre comment appeler la méthode `Camera.getCamera()` pour vérifier que l'utilisateur dispose d'au moins une caméra :

```
var cam:Camera = Camera.getCamera();
if (cam == null)
{
    trace("User has no cameras installed.");
}
else
{
    trace("User has at least 1 camera installed.");
}
```

Etant donné que la classe Camera n'étend pas la classe DisplayObject, il est impossible de l'ajouter directement à la liste d'affichage à l'aide de la méthode `addChild()`. Pour afficher le signal vidéo provenant de la caméra, vous devez donc créer un objet Video et appeler la méthode `attachCamera()` de l'occurrence de Video.

Ce fragment de code montre comment établir la connexion avec la caméra, s'il en existe une. Dans le cas contraire, Flash Player n'affiche rien :

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

Caméras des périphériques mobiles

La classe Camera n'est pas prise en charge dans le moteur d'exécution de Flash Player sur les navigateurs mobiles.

Dans les applications AIR des périphériques mobiles, vous pouvez accéder à la ou aux caméras dont dispose le périphérique. Sur les périphériques mobiles, vous pouvez utiliser soit la caméra frontale soit la caméra arrière, mais pas les deux en même temps. (Si vous activez une deuxième caméra, la première est tout d'abord désactivée.) La caméra frontale est mise en miroir horizontalement sur iOS ; elle ne l'est pas sur Android.

Détection de l'autorisation d'accéder à la caméra

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans le sandbox de l'application AIR, l'application peut accéder à n'importe quelle caméra sans autorisation de l'utilisateur. Sur Android, l'application doit toutefois spécifier l'autorisation CAMERA d'Android dans le descripteur d'application.

Avant que Flash Player ne puisse afficher les résultats d'une caméra, l'utilisateur doit autoriser Flash Player à accéder à la caméra de façon explicite. Lorsque la méthode `attachCamera()` est appelée, la boîte de dialogue Paramètres de Flash Player est affichée pour permettre à l'utilisateur d'autoriser ou refuser à Flash Player l'accès à la caméra et au microphone. Si l'utilisateur a accordé son autorisation, Flash Player affiche les résultats de la caméra dans l'occurrence de l'objet Video sur la scène. Si l'utilisateur a refusé, Flash Player ne peut pas se connecter à la caméra et l'objet Video n'affiche rien.

Pour déterminer si l'utilisateur a accordé à Flash Player l'accès à la caméra, vous pouvez écouter l'événement `status` de la caméra (`StatusEvent.STATUS`), comme dans le code suivant :

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
function statusHandler(event:StatusEvent):void
{
    // This event gets dispatched when the user clicks the "Allow" or "Deny"
    // button in the Flash Player Settings dialog box.
    trace(event.code); // "Camera.Muted" or "Camera.Unmuted"
}
```

La fonction `statusHandler()` est appelée dès que l'utilisateur clique sur Autoriser ou Refuser. Deux méthodes permettent de détecter le bouton qui a été cliqué :

- Le paramètre `event` de la fonction `statusHandler()` possède une propriété code qui contient la chaîne « Camera.Muted » ou « Camera.Unmuted ». Si la valeur est « Camera.Muted », l'utilisateur a refusé l'autorisation et Flash Player ne peut pas accéder à la caméra. Le fragment de code suivant en est un exemple :

  ```
  function statusHandler(event:StatusEvent):void
  {
      switch (event.code)
      {
          case "Camera.Muted":
              trace("User clicked Deny.");
              break;
          case "Camera.Unmuted":
              trace("User clicked Accept.");
              break;
      }
  }
  ```

- La classe Camera contient une propriété en lecture seule appelée `muted` qui indique si l'utilisateur a refusé l'accès à la caméra (`true`) ou l'a autorisé (`false`) dans le panneau Confidentialité de Flash Player. Le fragment de code suivant en est un exemple :

  ```
  function statusHandler(event:StatusEvent):void
  {
      if (cam.muted)
      {
          trace("User clicked Deny.");
      }
      else
      {
          trace("User clicked Accept.");
      }
  }
  ```

En consultant l'événement d'état à distribuer, vous pouvez rédiger du code pour gérer l'autorisation ou le refus d'accès à la caméra et terminer proprement. Par exemple, si l'utilisateur a refusé, vous pouvez afficher un message lui expliquant qu'il doit donner son autorisation s'il veut participer à une conversation vidéo, ou au contraire veiller à supprimer l'objet Video de la liste d'affichage afin de libérer des ressources système.

Dans AIR, l'autorisation d'utilisation de la caméra n'étant pas dynamique, un objet Camera ne distribue pas d'événement d'état.

Optimisation de la qualité des vidéos de la caméra

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, les nouvelles occurrences de la classe Video ont une largeur de 320 pixels sur une hauteur de 240 pixels. Pour optimiser la qualité vidéo, veillez à donner à vos objets Video les mêmes dimensions que celles de l'objet Camera. Pour obtenir la largeur et la hauteur de l'objet Camera, utilisez les propriétés `width` et `height` de la classe Camera. Vous pouvez alors adapter les propriétés `width` et `height` de l'objet Video à ces dimensions, ou transmettre celles-ci à la méthode constructeur de la classe Video, comme dans le fragment de code ci-dessous :

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

Comme la méthode `getCamera()` renvoie une référence à un objet Camera (ou `null` si aucune caméra n'est présente), vous pouvez utiliser les méthodes et les propriétés de cet objet même si l'utilisateur refuse l'accès à la caméra. Vous pouvez ainsi définir les dimensions de l'occurrence de l'objet Video en fonction de celles de la caméra.

```
var vid:Video;
var cam:Camera = Camera.getCamera();

if (cam == null)
{
    trace("Unable to locate available cameras.");
}
else
{
    trace("Found camera: " + cam.name);
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    vid = new Video();
    vid.attachCamera(cam);
}
function statusHandler(event:StatusEvent):void
{
    if (cam.muted)
    {
        trace("Unable to connect to active camera.");
    }
    else
    {
        // Resize Video object to match camera settings and
        // add the video to the display list.
        vid.width = cam.width;
        vid.height = cam.height;
        addChild(vid);
    }
    // Remove the status event listener.
    cam.removeEventListener(StatusEvent.STATUS, statusHandler);
}
```

Pour plus d'informations sur le mode plein écran, voir la section relative au mode plein écran sous « Définition des propriétés de la scène » à la page 171.

Gestion de l'état de la caméra

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Camera contient plusieurs propriétés qui permettent de suivre l'état de l'objet Camera. Par exemple, le code ci-dessous affiche plusieurs propriétés de la caméra à l'aide d'un objet Timer et d'une occurrence de champ de texte dans la liste d'affichage :

```
var vid:Video;
var cam:Camera = Camera.getCamera();
var tf:TextField = new TextField();
tf.x = 300;
tf.autoSize = TextFieldAutoSize.LEFT;
addChild(tf);

if (cam != null)
{
    cam.addEventListener(StatusEvent.STATUS, statusHandler);
    vid = new Video();
    vid.attachCamera(cam);
}
function statusHandler(event:StatusEvent):void
{
    if (!cam.muted)
    {
        vid.width = cam.width;
        vid.height = cam.height;
        addChild(vid);
        t.start();
    }
    cam.removeEventListener(StatusEvent.STATUS, statusHandler);
}

var t:Timer = new Timer(100);
t.addEventListener(TimerEvent.TIMER, timerHandler);
function timerHandler(event:TimerEvent):void
{
    tf.text = "";
    tf.appendText("activityLevel: " + cam.activityLevel + "\n");
    tf.appendText("bandwidth: " + cam.bandwidth + "\n");
    tf.appendText("currentFPS: " + cam.currentFPS + "\n");
    tf.appendText("fps: " + cam.fps + "\n");
    tf.appendText("keyFrameInterval: " + cam.keyFrameInterval + "\n");
    tf.appendText("loopback: " + cam.loopback + "\n");
    tf.appendText("motionLevel: " + cam.motionLevel + "\n");
    tf.appendText("motionTimeout: " + cam.motionTimeout + "\n");
    tf.appendText("quality: " + cam.quality + "\n");
}
```

Tous les 1/10 de seconde (100 millisecondes) l'événement `timer` de l'objet Timer est diffusé et la fonction `timerHandler()` actualise le contenu du champ de texte dans la liste d'affichage.

Chapitre 27 : Utilisation de la gestion des droits d'auteur numériques (DRM)

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Adobe® Access™ est une solution de protection du contenu. Elle permet aux propriétaires, aux distributeurs et aux publicitaires d'exploiter de nouvelles sources de revenus en assurant un accès transparent à un contenu Premium. Les éditeurs utilisent Adobe Access pour chiffrer du contenu, créer des stratégies et délivrer des licences. Adobe Flash Player et Adobe AIR intègrent une bibliothèque DRM, le module Adobe Access. Ce module permet au moteur d'exécution de communiquer avec le serveur de licences Adobe Access et de lire le contenu protégé. Le moteur d'exécution complète ainsi le cycle de vie du contenu protégé par Adobe Access et distribué par Flash Media Server.

Grâce à Adobe Access, les fournisseurs de contenu sont en mesure de proposer des contenus libres de droits et des contenus Premium. Imaginons qu'un consommateur souhaite visionner une émission de télévision sans coupure publicitaire. Il s'enregistre et s'acquitte d'un droit auprès de l'éditeur de contenu. Il peut ensuite entrer ses informations d'identification pour accéder au site et visionner l'émission sans publicité.

Prenons à présent l'exemple d'un consommateur qui souhaite visionner un contenu hors connexion lorsqu'il est en déplacement et ne dispose pas d'un accès à Internet. Ce flux de travail hors connexion est pris en charge par les applications AIR. Après s'être enregistré et acquitté des droits relatifs au contenu auprès de l'éditeur, l'utilisateur peut accéder au contenu et à l'application AIR associée et les télécharger à partir du site Web de l'éditeur. Par le biais de l'application AIR, l'utilisateur peut visionner le contenu hors connexion pendant la période autorisée. Il est également possible de partager le contenu avec d'autres périphériques dans le même groupe de périphériques à l'aide de domaines (**nouveauté dans la version 3.0**).

Adobe prend également en charge l'accès anonyme, qui ne nécessite pas d'authentification. Par exemple, un éditeur peut avoir recours à un accès anonyme pour distribuer du contenu publicitaire. Un accès anonyme permet en outre à un éditeur d'accéder gratuitement au contenu actuel pendant une période spécifique. Le fournisseur de contenu peut également stipuler et limiter le type et la version du lecteur requis pour le visionnement du contenu.

Lorsqu'un utilisateur tente de lire un contenu protégé dans Flash Player ou Adobe AIR, l'application doit appeler les API DRM. Les API DRM démarrent le flux de travail requis par la lecture du contenu protégé. Via le module Adobe Access, le moteur d'exécution contacte le serveur de licences. Le serveur de licences authentifie l'utilisateur, le cas échéant, et délivre une licence qui autorise la lecture du contenu protégé. Le moteur d'exécution reçoit la licence et déchiffre le contenu pour qu'il puisse être lu.

La procédure permettant à l'application de lire un contenu protégé par Adobe Access est décrite ci-après. Il est inutile de maîtriser le chiffrement de contenu ou la gestion des régulations par le biais d'Adobe Access. Nous partons toutefois du principe que vous communiquez avec le serveur de licences Adobe Access pour authentifier l'utilisateur et récupérer la licence. Nous considérons également comme acquis que vous créez une application réservée à la lecture de contenu protégé par Adobe Access.

Pour accéder à une présentation d'Adobe Access, notamment la création de régulations, voir la documentation livrée avec le produit.

Voir aussi

flash.net.drm package

flash.net.NetConnection

flash.net.NetStream

Présentation du flux de travail associé au contenu protégé

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Important : Flash Player 11.5 et les versions ultérieures intègrent le module Adobe Access ; l'étape de mise à jour (l'appel de `SystemUpdater.update(SystemUpdaterType.DRM)`) est donc inutile. Les navigateurs et plates-formes suivants sont concernés :

- Contrôle ActiveX Flash Player 11.5, pour toutes les plates-formes à l'exception d'Internet Explorer sous Windows 8 sur les processeurs Intel
- Plug-in Flash Player 11.5, pour tous les navigateurs
- Adobe AIR (version de bureau et version mobile)

L'étape de mise à jour est *toujours requise* dans les cas suivants :

- Internet Explorer sous Windows 8 sur les processeurs Intel
- Flash Player 11.4 et versions antérieures, à l'exception de Google Chrome 22 et des versions ultérieures (toutes les plates-formes) ou de Google Chrome 21 et des versions ultérieures (Windows)

Remarque : *vous pouvez toujours appeler* `SystemUpdater.update(SystemUpdaterType.DRM)` *sur un système doté de Flash Player 11.5 ou d'une version ultérieure, mais rien n'est téléchargé.*

Le flux de travail de haut niveau suivant indique comment une application peut récupérer et lire du contenu protégé. Il part du principe que l'application est conçue spécifiquement pour lire un contenu protégé par Adobe Access :

1. Récupérez les métadonnées du contenu.
2. Exécutez les mises à jour de Flash Player, s'il y a lieu.
3. Vérifiez si une licence est disponible localement. Si tel est le cas, chargez-la et passez à l'étape 7. Dans le cas contraire, passez à l'étape 4.
4. Vérifiez si l'authentification est obligatoire. Dans le cas contraire, passez à l'étape 7.
5. Si l'authentification est obligatoire, demandez les informations d'identification à l'utilisateur et transmettez-les au serveur de licences.
6. Si l'enregistrement de domaine est requis, joignez le domaine (AIR 3.0 et versions ultérieures).
7. Lorsque l'authentification aboutit, téléchargez la licence du serveur.
8. Lisez le contenu.

Si aucune erreur n'est survenue et si l'utilisateur a été autorisé à visionner le contenu, l'objet NetStream distribue un objet DRMStatusEvent. L'application procède alors à la lecture. L'objet DRMStatusEvent mémorise les informations relatives au voucher, qui identifient les régulations et autorisations de l'utilisateur. Il contient par exemple des informations relatives à la disponibilité du contenu hors connexion ou à la date d'expiration de la licence. L'application peut utiliser ces données pour avertir l'utilisateur de l'état de ses régulations. Elle peut par exemple afficher dans une barre d'état le nombre de jours restants pendant lesquels l'utilisateur est autorisé à visionner le contenu.

Si l'utilisateur est autorisé à accéder au contenu hors connexion, le voucher est mis en mémoire cache et le contenu chiffré est téléchargé sur la machine de l'utilisateur. L'utilisateur peut accéder au contenu pendant le nombre de jours défini dans la durée de mise en mémoire cache de la licence. La propriété `detail` de l'événement contient `"DRM.voucherObtained"`. L'application décide de l'emplacement local de stockage du contenu pour assurer sa disponibilité hors connexion. Vous pouvez également précharger les vouchers à l'aide de la classe DRMManager.

Remarque : *la mise en mémoire cache et le préchargement de vouchers sont pris en charge dans AIR et dans Flash Player. Néanmoins, le téléchargement et le stockage du contenu chiffré sont pris en charge uniquement dans AIR.*

C'est l'application elle-même qui est chargée de gérer explicitement les événements d'erreur, notamment lorsque l'utilisateur saisit correctement ses informations d'identification, mais que le voucher qui protège le contenu chiffré limite l'accès au contenu. Un utilisateur authentifié ne peut par exemple pas accéder au contenu s'il ne s'est pas acquitté des droits. Ce cas peut également se produire lorsque deux membres inscrits du même éditeur tentent de partager du contenu que seul l'un d'entre eux a payé. L'application doit informer l'utilisateur de l'erreur et proposer une solution de remplacement. Par exemple, l'application peut donner à l'utilisateur des instructions sur le mode d'inscription et de paiement des droits d'affichage.

Flux de travail détaillé des API

Flash Player 10.1 et les versions ultérieures, AIR 2.0 et les versions ultérieures

Ce flux de travail illustre de manière plus détaillée le flux de travail associé à un contenu protégé. Il décrit les API impliquées dans la lecture du contenu protégé par Adobe Access.

1. Par le biais d'un objet URLLoader, chargez les octets du fichier de métadonnées du contenu protégé. Définissez cet objet sur une variable, telle que `metadata_bytes`.

   Tout contenu contrôlé par Adobe Access contient des métadonnées Adobe Access. Lorsque le contenu est mis en package, ces métadonnées peuvent être enregistrées dans un fichier de métadonnées distinct (.metadata) parallèlement au contenu. Pour plus d'informations, voir la documentation d'Adobe Access.

2. Créez une occurrence de DRMContentData. Placez ce code dans un bloc try-catch :

   ```
   new DRMContentData(metadata_bytes)
   ```

   où *`metadata_bytes`* est l'objet URLLoader obtenu à l'étape 1.

3. (Flash Player uniquement) Le moteur d'exécution recherche le module Adobe Access. S'il ne le trouve pas, une erreur IllegalOperationError avec le code d'erreur DRMErrorEvent 3344 ou DRMErrorEvent 3343 est renvoyée.

   Pour gérer cette erreur, téléchargez le module Adobe Access à l'aide de l'API SystemUpdater. Une fois ce module téléchargé, l'objet SystemUpdater distribue un événement COMPLETE. Définissez un écouteur d'événement qui retourne à l'étape 2 lors de la distribution de cet événement. Le code suivant illustre ces étapes :

   ```
   flash.system.SystemUpdater.addEventListener(Event.COMPLETE, updateCompleteHandler);
   flash.system.SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
   ```

```
private function updateCompleteHandler (event:Event):void {
    /*redo step 2*/
    drmContentData = new DRMContentData(metadata_bytes);
}
```

Si le lecteur en tant que tel doit être mis à jour, un événement d'état est distribué. Pour plus d'informations sur la gestion de cet événement, voir « Ecoute d'un événement de mise à jour » à la page 562.

Remarque : *dans les applications AIR, le programme d'installation AIR gère la mise à jour du module Adobe Access et les mises à jour du moteur d'exécution requises.*

4 Créez des écouteurs qui écoutent les événements DRMStatusEvent et DRMErrorEvent distribués par l'objet DRMManager :

```
DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
```

Dans l'écouteur d'événements DRMStatusEvent, vérifiez que le voucher est valide (valeur autre que null). Dans l'écouteur d'événements DRMErrorEvent, gérez les événements DRMErrorEvents. Voir « Utilisation de la classe DRMStatusEvent » à la page 554 et « Utilisation de la classe DRMErrorEvent » à la page 559.

5 Chargez le voucher (la licence) requis pour lire le contenu.

Essayez tout d'abord de charger une licence stockée localement pour lire le contenu :

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

Une fois le chargement terminé, l'objet DRMManager distribue `DRMStatusEvent.DRM_Status`.

6 Si l'objet DRMVoucher possède une valeur autre que null, le voucher est valide. Passez à l'étape 13.

7 Si l'objet DRMVoucher est défini sur null, vérifiez la méthode d'authentification requise par la régulation associée au contenu. Faites appel à la propriété `DRMContentData.authenticationMethod`.

8 Si la méthode d'authentification est `ANONYMOUS`, passez à l'étape 13.

9 Si la méthode d'authentification est `USERNAME_AND_PASSWORD`, l'application doit intégrer un mécanisme permettant à l'utilisateur d'entrer des informations d'identification. Transmettez ces informations d'identification au serveur de licences pour authentifier l'utilisateur :

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

DRMManager distribue un événement `DRMAuthenticationErrorEvent` si l'authentification échoue, un événement `DRMAuthenticationCompleteEvent` si elle aboutit. Associez des écouteurs à ces événements.

10 Si la méthode d'authentification est `UNKNOWN`, une méthode d'authentification personnalisée doit être utilisée. Dans ce cas, le fournisseur de contenu a prévu d'exécuter l'authentification hors bande, c'est-à-dire de ne pas utiliser les API ActionScript 3.0. La procédure d'authentification personnalisée doit produire un jeton d'authentification qu'il est possible de transmettre à la méthode `DRMManager.setAuthenticationToken()`.

11 Si l'authentification échoue, l'application doit retourner à l'étape 9. Assurez-vous que l'application intègre un mécanisme permettant de gérer et restreindre les échecs successifs d'authentification. Après trois tentatives, vous pouvez, par exemple, afficher un message indiquant à l'utilisateur que l'authentification a échoué et qu'il est impossible de lire le contenu.

12 Pour utiliser le jeton stocké au lieu d'inviter l'utilisateur à entrer des informations d'identification, définissez-le à l'aide de la méthode `DRMManager.setAuthenticationToken()`. Téléchargez ensuite la licence à partir du serveur de licences et lisez le contenu (voir étape 8).

13 (Facultatif) Si l'authentification aboutit, vous pouvez capturer le jeton d'authentification, à savoir un tableau d'octets placé en mémoire cache. Récupérez ce jeton à l'aide de la propriété `DRMAuthenticationCompleteEvent.token`. Vous pouvez stocker et utiliser le jeton d'authentification pour éviter à l'utilisateur d'avoir à entrer plusieurs fois les informations d'identification associées au contenu. Le serveur de licences détermine la période de validité du jeton d'authentification.

14 Si l'authentification aboutit, téléchargez la licence du serveur de licences :

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

Une fois le chargement terminé, l'objet DRMManager distribue l'événement DRMStatusEvent.DRM_STATUS. Ecoutez cet événement. Une fois ce dernier distribué, vous pouvez lire le contenu.

15 Lisez la vidéo en créant un objet NetStream, puis en appelant sa méthode `play()` :

```
stream = new NetStream(connection);
stream.addEventListener(DRMStatusEvent.DRM _STATUS, drmStatusHandler);
stream.addEventListener(DRMErrorEvent.DRM_ERROR, drmErrorHandler);
stream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
stream.client = new CustomClient();
video.attachNetStream(stream);
stream.play(videoURL);
```

Objets DRMContentData et objets session

Lors de la création d'un objet `DRMContentData`, celui-ci est utilisé en tant qu'objet session qui fait référence au module DRM de Flash Player. Toutes les API `DRMManager` qui reçoivent cet objet `DRMContentData` utilisent ce module DRM particulier. Il existe néanmoins deux API `DRMManager` qui n'utilisent pas l'objet `DRMContentData`. Voici ces règles :

1 `authenticate()`

2 `setAuthenticationToken()`

Etant donné qu'il n'existe aucun objet `DRMContentData` associé, l'invocation de ces API `DRMManager` fait appel au module DRM le plus récent du disque. Cela peut être problématique si une mise à jour du module DRM se produit au cours du flux de travail du DRM de l'application. Tenez compte du scénario suivant :

1 L'application crée un objet `DRMContentDatacontentData1`, qui utilise *AdobeCP1* comme module DRM.

2 L'application invoque la méthode `DRMManager.authenticate(contentData1.serverURL,...)`.

3 L'application invoque la méthode `DRMManager.loadVoucher(contentData1, ...)`.

Si une mise à jour du module DRM se produit avant que l'application ne puisse accéder à l'étape 2, la méthode `DRMManager.authenticate()` finit par effectuer l'authentification à l'aide du module DRM *AdobeCP2*. La méthode `loadVoucher()` de l'étape 3 échoue, car elle utilise toujours le module DRM *AdobeCP1*. Il est possible que la mise à jour ait eu lieu suite à l'invocation de la mise à jour du module DRM par une autre application. Vous pouvez éviter ce scénario en invoquant la mise à jour du module DRM au démarrage de l'application.

Evénements DRM

Le moteur d'exécution distribue un grand nombre d'événements lorsqu'une application essaie de lire un contenu protégé :

- DRMDeviceGroupErrorEvent (AIR uniquement), distribué par DRMManager
- DRMAuthenticateEvent (AIR uniquement), distribué par NetStream
- DRMAuthenticationCompleteEvent, distribué par DRMManager
- DRMAuthenticationErrorEvent, distribué par DRMManager

- DRMErrorEvent, distribué par NetStream et DRMManager
- DRMStatusEvent, distribué par NetStream et DRMManager
- StatusEvent
- NetStatusEvent (voir « Ecoute d'un événement de mise à jour » à la page 562)

Pour prendre en charge un contenu protégé par Adobe Access, associez des écouteurs aux événements DRM.

Préchargement de vouchers pour une lecture hors connexion

Adobe AIR 1.5 et les versions ultérieures

Vous pouvez précharger les vouchers (licences) requis pour lire le contenu protégé par Adobe Access. Les vouchers préchargés permettent aux utilisateurs de visionner le contenu même si leur connexion Internet n'est pas active. (Le processus de préchargement à proprement parler requiert une connexion à Internet.) Faites appel à la méthode `preloadEmbeddedMetadata()` de la classe NetStream et à la classe DRMManager pour précharger les vouchers. Dans AIR 2.0 et les versions ultérieures, vous pouvez précharger directement des vouchers par le biais d'un objet DRMContentData. Il est préférable d'utiliser cette technique, car elle permet de mettre à jour l'objet DRMContentData indépendamment du contenu. (La méthode `preloadEmbeddedData()` extrait l'objet DRMContentData du contenu.)

Utilisation de l'objet DRMContentData

Adobe AIR 2.0 et les versions ultérieures

La procédure ci-dessous décrit le préchargement du voucher associé à un fichier multimédia protégé par le biais d'un objet DRMContentData.

1. Extrayez les métadonnées binaires associées au contenu mis en package. Si vous utilisez Adobe Access Java Reference Packager, ce fichier de métadonnées est automatiquement généré avec une extension *.metadata*. Vous pourriez, par exemple, télécharger ces métadonnées à l'aide de la classe URLLoader.
2. Créez un objet DRMContentData en transmettant les métadonnées à la fonction constructeur :

   ```
   var drmData:DRMContentData = new DRMContentData( metadata );
   ```

3. Les autres étapes sont identiques au flux de travail décrit à la section « Présentation du flux de travail associé au contenu protégé » à la page 546.

Utilisation de preloadEmbeddedMetadata()

Adobe AIR 1.5 et les versions ultérieures

La procédure suivante décrit le préchargement du voucher associé à un fichier multimédia protégé par DRM par le biais de `preloadEmbeddedMetadata()` :

1. Téléchargez et stockez le fichier multimédia. (Il est uniquement possible de précharger les métadonnées du module DRM à partir des fichiers enregistrés localement.)
2. Créez les objets NetConnection et NetStream, en fournissant des implémentations pour les fonctions de rappel `onDRMContentData()` et `onPlayStatus()` de l'objet client NetStream.
3. Créez un objet NetStreamPlayOptions et définissez la propriété `stream` sur l'URL du fichier multimédia local.
4. Appelez la méthode NetStream `preloadEmbeddedMetadata()`, en transmettant l'objet NetStreamPlayOptions identifiant le fichier multimédia à analyser.

5 Si le fichier multimédia contient des métadonnées DRM, la fonction de rappel `onDRMContentData()` est invoquée. Les métadonnées sont transmises à cette fonction sous forme d'objet DRMContentData.

6 Utilisez l'objet DRMContentData pour obtenir le voucher à l'aide de la méthode DRMManager `loadVoucher()`.

Si la valeur de la propriété `authenticationMethod` de l'objet `DRMContentData` est `flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD`, authentifiez l'utilisateur sur le serveur de droits multimédias avant de charger le voucher. Les propriétés `serverURL` et `domain` de l'objet DRMContentData peuvent être transmis à la méthode DRMManager `authenticate()`, de même que les informations d'identification de l'utilisateur.

7 La fonction de rappel `onPlayStatus()` est invoquée lorsque l'analyse du fichier est terminée. Si la fonction `onDRMContentData()` n'a pas été appelée, le fichier ne contient pas les métadonnées nécessaires à l'obtention d'un voucher. L'absence de cet appel peut également signifier qu'Adobe Access ne protège pas ce fichier.

L'exemple de code pour AIR suivant illustre le préchargement d'un voucher associé à un fichier multimédia local :

```
package
{
import flash.display.Sprite;
import flash.events.DRMAuthenticationCompleteEvent;
import flash.events.DRMAuthenticationErrorEvent;
import flash.events.DRMErrorEvent;
import flash.ev ents.DRMStatusEvent;
import flash.events.NetStatusEvent;
import flash.net.NetConnection;
import flash.net.NetStream;
import flash.net.NetStreamPlayOptions;
import flash.net.drm.AuthenticationMethod;
import flash.net.drm.DRMContentData;
import flash.net.drm.DRMManager;
import flash.net.drm.LoadVoucherSetting;
public class DRMPreloader extends Sprite
{
    private var videoURL:String = "app-storage:/video.flv";
    private var userName:String = "user";
    private var password:String = "password";
    private var preloadConnection:NetConnection;
    private var preloadStream:NetStream;
    private var drmManager:DRMManager = DRMManager.getDRMManager();
    private var drmContentData:DRMContentData;
    public function DRMPreloader():void {
        drmManager.addEventListener(
            DRMAuthenticationCompleteEvent.AUTHENTICATION_COMPLETE,
            onAuthenticationComplete);
        drmManager.addEventListener(DRMAuthenticationErrorEvent.AUTHENTICATION_ERROR,
            onAuthenticationError);
        drmManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
        drmManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
        preloadConnection = new NetConnection();
        preloadConnection.addEventListener(NetStatusEvent.NET_STATUS, onConnect);
        preloadConnection.connect(null);
    }

    private function onConnect( event:NetStatusEvent ):void
    {
        preloadMetadata();
    }
```

```
    private function preloadMetadata():void
    {
        preloadStream = new NetStream( preloadConnection );
        preloadStream.client = this;
        var options:NetStreamPlayOptions = new NetStreamPlayOptions();
        options.streamName = videoURL;
        preloadStream.preloadEmbeddedData( options );
    }
    public function onDRMContentData( drmMetadata:DRMContentData ):void
    {
        drmContentData = drmMetadata;
        if ( drmMetadata.authenticationMethod == AuthenticationMethod.USERNAME_AND_PASSWORD )
        {
            authenticateUser();
        }
        else
            {
                getVoucher();
            }
    }
    private function getVoucher():void
    {
        drmManager.loadVoucher( drmContentData, LoadVoucherSetting.ALLOW_SERVER );
    }

    private function authenticateUser():void
    {
        drmManager.authenticate( drmContentData.serverURL, drmContentData.domain, userName,
password );
    }
    private function onAuthenticationError( event:DRMAuthenticationErrorEvent ):void
    {
        trace( "Authentication error: " + event.errorID + ", " + event.subErrorID );
    }

    private function onAuthenticationComplete( event:DRMAuthenticationCompleteEvent ):void
    {
        trace( "Authenticated to: " + event.serverURL + ", domain: " + event.domain );
        getVoucher();
    }
    private function onDRMStatus( event:DRMStatusEvent ):void
    {
        trace( "DRM Status: " + event.detail);
        trace("--Voucher allows offline playback = " + event.isAvailableOffline );
        trace("--Voucher already cached          = " + event.isLocal );
        trace("--Voucher required authentication = " + !event.isAnonymous );
    }
    private function onDRMError( event:DRMErrorEvent ):void
    {
        trace( "DRM error event: " + event.errorID + ", " + event.subErrorID + ", " + event.text );
    }
    public function onPlayStatus( info:Object ):void
    {
        preloadStream.close();
    }
}
}
```

Membres et événements DRM de la classe NetStream

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe NetStream propose une connexion en flux continu unidirectionnelle entre Flash Player ou une application AIR et Flash Media Server ou le système de fichiers local. (La classe NetStream gère également les téléchargements en mode progressif.) Un objet NetStream est un canal dans un objet NetConnection. La classe NetStream distribue quatre événements DRM :

Evénement	Description
drmAuthenticate (AIR uniquement)	Défini dans la classe DRMAuthenticateEvent. Cet événement est distribué lorsqu'un objet NetStream tente de lire un contenu protégé qui requiert l'authentification des informations d'identification de l'utilisateur avant la lecture. Les propriétés de cet événement incluent les propriétés header, usernamePrompt, passwordPrompt et urlPrompt, qui permettent d'obtenir et de définir les informations d'identification des utilisateurs. Cet événement est répété jusqu'à ce que l'objet NetStream reçoive des informations d'identification des utilisateurs valides.
drmError	Défini dans la classe DRMErrorEvent et distribué lorsqu'un objet NetStream essaie de lire un contenu protégé et rencontre une erreur DRM. Un objet d'événement associé à une erreur DRM est par exemple distribué lorsque l'autorisation de l'utilisateur échoue. Cette erreur s'est peut-être produite car l'utilisateur ne s'est pas acquitté des droits d'affichage du contenu. Il est également possible que le fournisseur de contenu ne prenne pas en charge l'application d'affichage.
drmStatus	Défini dans la classe DRMStatusEvent. Cet événement est distribué lorsque la lecture du contenu protégé démarre (une fois l'utilisateur authentifié et autorisé à lire le contenu). L'objet DRMStatusEvent contient des informations relatives au voucher. Il mémorise par exemple des informations relatives à la disponibilité hors connexion du contenu ou à la date d'expiration du voucher (au terme de laquelle il est impossible de visionner le contenu).
status	Défini dans l'événement events.StatusEvent et distribué uniquement lorsque l'application essaie de lire un contenu protégé, en appelant la méthode NetStream.play(). La valeur de la propriété status code est définie sur « DRM.encryptedFLV ».

La classe NetStream comprend les méthodes propres à DRM suivantes, réservées à AIR :

Méthode	Description
resetDRMVouchers()	Supprime toutes les données du voucher DRM placé dans le cache local. L'application doit télécharger à nouveau les vouchers pour que l'utilisateur puisse accéder au contenu chiffré. Par exemple, le code suivant supprime tous les vouchers du cache : NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	Transmet un jeu d'informations d'identification, à savoir le nom d'utilisateur, le mot de passe et le type d'authentification, à l'objet NetStream à titre d'authentification. Les types d'authentification valides sont « `drm` » et « `proxy` ». Pour le type d'authentification « `drm` », les informations d'identification indiquées sont comparées à Adobe Access. Pour le type d'authentification « `proxy` », les informations d'identification sont comparées aux données stockées sur le serveur proxy et doivent être identiques aux informations requises par ce dernier. Par exemple, une entreprise peut solliciter l'authentification de l'application auprès d'un serveur proxy avant que l'utilisateur puisse accéder à Internet. L'option proxy permet également ce type d'authentification. A moins que l'authentification anonyme ne soit utilisée, au terme de l'authentification proxy, l'utilisateur doit néanmoins s'authentifier auprès d'Adobe Access pour obtenir le voucher et lire le contenu. Vous pouvez utiliser `setDRMAuthenticationCredentials()` une deuxième fois, avec l'option « `drm` », pour l'authentification auprès d'Adobe Access.
preloadEmbeddedMetadata()	Recherche les métadonnées intégrées dans un fichier multimédia local. Lorsque des métadonnées DRM sont détectées, AIR appelle la fonction de rappel `onDRMContentData()`.

Par ailleurs, dans AIR, un objet NetStream appelle les fonctions de rappel `onDRMContentData()` et `onPlayStatus()` suite à un appel à la méthode `preloadEmbeddedMetaData()`. La fonction `onDRMContentData()` est appelée lorsque des métadonnées DRM sont détectées dans un fichier multimédia. La fonction `onPlayStatus()` est appelée une fois l'analyse du fichier terminée. Les fonctions `onDRMContentData()` et `onPlayStatus()` doivent être définies sur l'objet `client` affecté à l'occurrence de NetStream. Si vous utilisez le même objet NetStream pour précharger les vouchers et lire un contenu, attendez l'appel `onPlayStatus()` généré par `preloadEmbeddedMetaData()` avant de commencer la lecture.

Dans le code suivant pour AIR, le nom d'utilisateur (« administrator »), le mot de passe (« password ») et le type d'authentification (« drm ») sont définis pour authentifier l'utilisateur. La méthode setDRMAuthenticationCredentials() doit fournir des informations d'identification connues et acceptées par le fournisseur de contenu. Ces informations d'identification sont identiques aux données saisies par l'utilisateur pour visionner le contenu. Ce chapitre ne contient pas le code permettant de lire la vidéo et de s'assurer qu'une connexion au flux vidéo a abouti.

```
var connection:NetConnection = new NetConnection();
connection.connect(null);

var videoStream:NetStream = new NetStream(connection);

videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                             drmAuthenticateEventHandler)

private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
{
    videoStream.setDRMAuthenticationCredentials("administrator", "password", "drm");
}
```

Utilisation de la classe DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 et les versions ultérieures

Un objet NetStream distribue un objet DRMStatusEvent lorsque la lecture du contenu protégé par Adobe Access débute. (Pour cela, la licence doit être vérifiée, et l'utilisateur doit être authentifié et autorisé à afficher le contenu). L'objet DRMStatusEvent est également distribué lorsqu'un utilisateur anonyme est autorisé à accéder au contenu. La licence est vérifiée pour s'assurer que les utilisateurs anonymes, qui ne requièrent pas d'authentification, sont autorisés à lire le contenu. Les utilisateurs anonymes risquent de ne pas être autorisés à accéder au contenu pour diverses raisons. Un utilisateur anonyme ne peut, par exemple, pas accéder au contenu lorsque la licence est arrivée à expiration.

L'objet DRMStatusEvent contient des informations relatives à la licence. Il mémorise par exemple des informations relatives à la disponibilité hors connexion de la licence ou à la date d'expiration du voucher (au terme de laquelle il est impossible de visionner le contenu). L'application peut utiliser ces données pour communiquer l'état des régulations de l'utilisateur et les autorisations correspondantes.

Propriétés DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 et les versions ultérieures

La classe DRMStatusEvent comprend les propriétés suivantes. Certaines propriétés ont été introduites dans les versions d'AIR ultérieures à 1.0. Pour obtenir des informations de version détaillées, voir le *Guide de référence d'ActionScript 3.0 pour Flash Professional*.

La classe DRMVoucher propose des propriétés pour Flash Player 10.1 similaires à celles qui ne sont pas prises en charge par ce dernier.

Propriété	Description
contentData	Objet DRMContentData contenant les métadonnées DRM intégrées dans le contenu.
detail (AIR uniquement)	Chaîne expliquant le contexte de l'événement d'état. Dans DRM 1.0, l'unique valeur valide est DRM.voucherObtained.
isAnonymous (AIR uniquement)	Indique si le contenu protégé avec Adobe Access est disponible sans que l'utilisateur n'ait à fournir ses informations d'authentification (true), ou uniquement à condition qu'il fournisse ces informations (false). La valeur false signifie que l'utilisateur doit entrer un nom d'utilisateur et un mot de passe correspondant aux données connues et attendues par le fournisseur de contenu.
isAvailableOffline (AIR uniquement)	Indique si le contenu protégé avec Adobe Access peut être disponible hors connexion (true) ou pas (false). Pour que le contenu protégé numériquement soit disponible hors ligne, le voucher correspondant doit être placé dans le cache de l'ordinateur local de l'utilisateur.
isLocal	Indique si le voucher requis pour la lecture du contenu est mis en cache localement.
offlineLeasePeriod (AIR uniquement)	Nombre restant de jours pendant lesquels le contenu peut être visionné hors ligne.
policies (AIR uniquement)	Objet personnalisé pouvant contenir des propriétés DRM personnalisées.
voucher	DRMVoucher.
voucherEndDate (AIR uniquement)	Date absolue d'expiration du voucher, après laquelle il est impossible de visionner le contenu.

Création d'un gestionnaire DRMStatusEvent

Flash Player 10.1, Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée un gestionnaire d'événement qui renvoie des informations sur l'état du contenu chiffré par DRM de l'objet NetStream à l'origine de l'événement. Ajoutez ce gestionnaire d'événement à un objet NetStream qui pointe vers le contenu protégé.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

Utilisation de la classe DRMAuthenticateEvent

Adobe AIR 1.0 et les versions ultérieures

L'objet DRMAuthenticateEvent est distribué lorsqu'un objet NetStream essaie de lire un contenu protégé qui requiert la saisie des informations d'identification de l'utilisateur avant la lecture.

Le gestionnaire DRMAuthenticateEvent est chargé de collecter les informations d'identification requises (nom d'utilisateur, mot de passe et type) et de les transmettre à la méthode `NetStream.setDRMAuthenticationCredentials()` pour qu'elles soient validées. Chaque application AIR doit intégrer un mécanisme de collecte des informations d'identification des utilisateurs. L'application pourrait par exemple proposer à l'utilisateur une interface simple permettant de saisir son nom d'utilisateur et son mot de passe. Veillez également à intégrer un mécanisme de gestion et de restriction des tentatives d'authentification successives.

Propriétés DRMAuthenticateEvent

Adobe AIR 1.0 et les versions ultérieures

La classe DRMAuthenticateEvent comprend les propriétés suivantes :

Propriété	Description
authenticationType	Indique si les informations d'identification fournies sont destinées à une authentification auprès d'Adobe Access (« drm ») ou d'un serveur proxy (« proxy »). Par exemple, l'option "proxy" permet à l'application de s'authentifier auprès d'un serveur proxy, s'il y a lieu, avant que l'utilisateur puisse accéder à Internet. A moins que l'authentification anonyme ne soit utilisée, au terme de l'authentification proxy, l'utilisateur doit néanmoins s'authentifier auprès d'Adobe Access pour obtenir le voucher et lire le contenu. Vous pouvez utiliser setDRMAuthenticationcredentials() une deuxième fois, avec l'option « drm », pour l'authentification auprès d'Adobe Access.
header	En-tête du fichier de contenu chiffré fourni par le serveur. Il contient des informations relatives au contexte du contenu chiffré. Il est possible de transmettre cette chaîne d'en-tête à l'application Flash pour autoriser cette dernière à créer une boîte de dialogue nom d'utilisateur/mot de passe. La chaîne d'en-tête peut être utilisée comme instructions de la boîte de dialogue. Par exemple, l'en-tête peut indiquer « Saisissez votre nom d'utilisateur et votre mot de passe ».
netstream	Objet NetStream à l'origine de cet événement.
passwordPrompt	Invite associée au mot de passe, fournie par le serveur. La chaîne peut comporter des instructions relatives au type de mot de passe requis.
urlPrompt	Invite associée à une chaîne URL, fournie par le serveur. La chaîne peut fournir l'emplacement auquel le nom d'utilisateur et le mot de passe sont envoyés.
usernamePrompt	Invite associée au nom d'utilisateur, fournie par le serveur. La chaîne peut comporter des instructions relatives au type de nom d'utilisateur requis. Par exemple, un fournisseur de contenu peut exiger une adresse électronique comme nom d'utilisateur.

Les chaînes mentionnées précédemment sont fournies par le serveur FMRMS uniquement. Adobe Access Server n'utilise pas ces chaînes.

Création d'un gestionnaire DRMAuthenticateEvent

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée un gestionnaire d'événement qui transmet un jeu d'informations d'authentification codées en dur à l'objet NetStream à l'origine de l'événement. (Ce chapitre ne contient pas le code permettant de lire la vidéo et de s'assurer qu'une connexion au flux vidéo a abouti.)

```
var connection:NetConnection = new NetConnection();
connection.connect(null);

var videoStream:NetStream = new NetStream(connection);

videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                             drmAuthenticateEventHandler)

private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
{
    videoStream.setDRMAuthenticationCredentials("administrator", "password", "drm");
}
```

Création d'une interface de collecte des informations d'identification des utilisateurs

Adobe AIR 1.0 et les versions ultérieures

Si un contenu protégé requiert l'authentification de l'utilisateur, l'application AIR doit généralement extraire les informations d'identification de l'utilisateur au moyen d'une interface utilisateur.

Le code suivant est un exemple Flex d'interface utilisateur simple permettant de collecter les informations d'identification des utilisateurs. Il se compose d'un objet Panel contenant deux objets TextInput (le premier étant réservé aux noms d'utilisateur et le second aux mots de passe). L'objet Panel comporte également un bouton permettant de démarrer la méthode `credentials()`.

```
<mx:Panel x="236.5" y="113" width="325" height="204" layout="absolute" title="Login">
    <mx:TextInput x="110" y="46" id="uName"/>
    <mx:TextInput x="110" y="76" id="pWord" displayAsPassword="true"/>
    <mx:Text x="35" y="48" text="Username:"/>
    <mx:Text x="35" y="78" text="Password:"/>
    <mx:Button x="120" y="115" label="Login" click="credentials()"/>
</mx:Panel>
```

La méthode `credentials()` est une méthode définie par l'utilisateur qui transmet les valeurs de nom d'utilisateur et de mot de passe à la méthode `setDRMAuthenticationCredentials()`. Une fois ces valeurs transmises, la méthode `credentials()` réinitialise les valeurs des objets TextInput.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

Pour implémenter ce type d'interface simple, il est possible d'inclure l'objet Panel au sein d'un nouvel état. Le nouvel état provient de l'état de base lorsque l'objet DRMAuthenticateEvent est envoyé. L'exemple suivant contient un objet VideoDisplay doté d'un attribut source qui pointe vers un fichier FLV protégé. Dans ce cas, la méthode `credentials()` est modifiée de façon à ce qu'elle renvoie également l'application à son état de base. Pour cela, les informations d'identification de l'utilisateur doivent être transmises et les valeurs de l'objet TextInput réinitialisées.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute"
    width="800"
    height="500"
    title="DRM FLV Player"
    creationComplete="initApp()" >

    <mx:states>
        <mx:State name="LOGIN">
            <mx:AddChild position="lastChild">
                    <mx:Panel x="236.5" y="113" width="325" height="204" layout="absolute"
                            title="Login">
                    <mx:TextInput x="110" y="46" id="uName"/>
                    <mx:TextInput x="110" y="76" id="pWord" displayAsPassword="true"/>
                    <mx:Text x="35" y="48" text="Username:"/>
                    <mx:Text x="35" y="78" text="Password:"/>
                    <mx:Button x="120" y="115" label="Login" click="credentials()"/>
                    <mx:Button x="193" y="115" label="Reset" click="uName.text='';
                            pWord.text='';"/>
                </mx:Panel>
            </mx:AddChild>
        </mx:State>
    </mx:states>

    <mx:Script>
        <![CDATA[
                import flash.events.DRMAuthenticateEvent;
            private function initApp():void
            {
                videoStream.addEventListener(DRMAuthenticateEvent.DRM_AUTHENTICATE,
                                    drmAuthenticateEventHandler);
            }
```

```
            public function credentials():void
            {
                videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
                uName.text = "";
                pWord.text = "";
                currentState='';
            }

            private function drmAuthenticateEventHandler(event:DRMAuthenticateEvent):void
            {
                currentState='LOGIN';
            }
        ]]>
    </mx:Script>

    <mx:VideoDisplay id="video" x="50" y="25" width="700" height="350"
        autoPlay="true"
        bufferTime="10.0"
        source="http://www.example.com/flv/Video.flv" />
</mx:WindowedApplication>
```

Utilisation de la classe DRMErrorEvent

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Adobe Flash Player et Adobe AIR distribuent un objet DRMErrorEvent lorsqu'un objet NetStream qui tente de lire un contenu protégé rencontre une erreur DRM. Si les informations d'identification ne sont pas valides dans une application AIR, l'objet DRMAuthenticateEvent distribue continuellement un objet jusqu'à ce que l'utilisateur entre des informations d'identification valides ou que l'application refuse toute autre tentative. L'application est chargée d'écouter tout autre événement d'erreur DRM pour détecter, identifier et gérer les erreurs DRM.

Même si les informations d'identification de l'utilisateur sont valides, il est possible que les conditions du voucher l'empêchent d'afficher un contenu chiffré. Par exemple, un utilisateur peut se voir refuser l'accès à un contenu s'il tente de lire ce contenu dans une application non autorisée. Une application non autorisée est une application que l'éditeur du contenu chiffré n'a pas validée. Dans ce cas, un objet DRMErrorEvent est distribué.

Des événements d'erreur peuvent également être déclenchés si le contenu est endommagé ou si la version de l'application ne correspond pas aux spécifications du voucher. L'application doit intégrer un mécanisme de gestion des erreurs approprié.

Propriétés DRMErrorEvent

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour obtenir la liste complète des erreurs, voir Erreurs d'exécution dans le Guide de référence d'ActionScript 3.0 pour Flash Professional. Les erreurs DRM débutent au numéro 3300.

Création d'un gestionnaire DRMErrorEvent

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée un gestionnaire d'événement associé à l'objet NetStream à l'origine de l'événement. Il est appelé si l'objet NetStream rencontre une erreur lors d'une tentative de lecture de contenu protégé. En règle générale, lorsqu'une application détecte une erreur, elle procède à une série d'opérations de nettoyage. Elle informe ensuite l'utilisateur de l'erreur et lui fournit des solutions pour résoudre le problème.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

Utilisation de la classe DRMManager

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

La classe DRMManager permet de gérer les vouchers et les sessions du serveur des droits multimédias dans les applications.

Gestion des vouchers (AIR uniquement)

Lorsqu'un utilisateur lit un contenu protégé, le moteur d'exécution obtient et place en mémoire cache la licence requise pour visionner le contenu. Si l'application enregistre le fichier localement et si la licence autorise la lecture hors connexion, l'utilisateur peut visionner le contenu dans l'application AIR. La lecture locale hors connexion est possible même si la connexion au serveur de droits multimédias n'est pas disponible. Vous pouvez pré-placer en mémoire cache le voucher par le biais de DRMManager et de la méthode `preloadEmbeddedMetadata()` de NetStream. Il est inutile que l'application obtienne la licence nécessaire au visionnement du contenu. Par exemple, votre application peut télécharger le fichier multimédia, puis obtenir le voucher pendant que l'utilisateur est encore en ligne.

Pour précharger un voucher, utilisez la méthode NetStream `preloadEmbeddedMetadata()` pour obtenir un objet DRMContentData. L'objet DRMContentData contient l'URL et le domaine du serveur de gestion des droits d'auteur pouvant fournir la licence et indique si l'authentification de l'utilisateur est requise. Avec ces informations, vous pouvez appeler la méthode DRMManager `loadVoucher()` pour obtenir le voucher et le mettre en cache. La procédure de préchargement des vouchers est décrite plus en détails à la section « Préchargement de vouchers pour une lecture hors connexion » à la page 550.

Gestion des sessions

Vous pouvez également utiliser DRMManager pour authentifier l'utilisateur auprès d'un serveur de gestion des droits d'auteur et pour gérer les sessions persistantes.

Appelez la méthode DRMManager `authenticate()` pour établir une session auprès du serveur de gestion des droits d'auteur. Une fois l'authentification réussie, le DRMManager distribue un objet DRMAuthenticationCompleteEvent qui contient un jeton de session. Vous pouvez enregistrer ce jeton pour établir les sessions futures de sorte que l'utilisateur n'ait plus besoin de fournir les informations d'identification de son compte. Transmettez le jeton à la méthode `setAuthenticationToken()` pour établir une nouvelle session authentifiée. (Les paramètres du serveur générés par le jeton déterminent la date d'expiration du jeton et d'autres attributs. Le code de l'application AIR ne doit pas interpréter la structure de données du jeton, car cette structure est susceptible de changer lors de futures mises à jour d'AIR.)

Il est possible de transférer les jetons d'authentification à d'autres ordinateurs. Pour les protéger, vous pouvez les stocker dans le magasin local chiffré d'AIR. Pour plus d'informations, voir « Stockage local chiffré » à la page 737.

Evénements DRMStatus

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le DRMManager distribue un objet DRMStatusEvent lorsqu'un appel de la méthode `loadVoucher()` aboutit.

Si un voucher a été obtenu, la propriété `detail` (AIR uniquement) de l'objet d'événement prend la valeur : « DRM.voucherObtained » et la propriété `voucher` contient l'objet DRMVoucher.

Si le voucher n'a pas été obtenu, la propriété `detail` (AIR uniquement) conserve la valeur : « DRM.voucherObtained », mais la propriété `voucher` est `null`. Il est impossible d'obtenir un voucher si, par exemple, vous définissez LoadVoucherSetting sur *localOnly* et qu'aucun voucher n'a été placé dans la mémoire cache locale.

Si l'appel de la méthode `loadVoucher()` échoue, en raison d'une erreur de communication ou d'authentification, par exemple, le DRMManager distribue alors un objet DRMErrorEvent ou DRMAuthenticationErrorEvent.

Evénements DRMAuthenticationComplete

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le DRMManager distribue un objet DRMAuthenticationCompleteEvent lorsque l'utilisateur est bien authentifié via un appel à la méthode `authenticate()`.

L'objet DRMAuthenticationCompleteEvent contient un jeton réutilisable qui peut servir à maintenir l'authentification de l'utilisateur dans plusieurs sessions de l'application. Transmettez ce jeton à la méthode `setAuthenticationToken()` du DRMManager pour rétablir la session. (Le créateur du jeton définit les attributs du jeton, notamment sa date d'expiration. Adobe ne fournit pas d'API capable d'examiner les attributs du jeton.)

Evénements DRMAuthenticationError

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Le DRMManager distribue un objet DRMAuthenticationErrorEvent lorsqu'un utilisateur n'a pas pu s'authentifier via un appel aux méthodes `authenticate()` ou `setAuthenticationToken()`.

Utilisation de la classe DRMContentData

Flash Player 10.1 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

L'objet DRMContentData contient les propriétés de métadonnées du contenu protégé par Adobe Access. Les propriétés DRMContentData contiennent les informations nécessaires à l'obtention du voucher de licence permettant d'afficher le contenu. La classe DRMContentData permet d'obtenir le fichier de métadonnées associé au contenu, comme indiqué à la section « Flux de travail détaillé des API » à la page 547.

Pour plus d'informations, voir la classe DRMContentData dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Mise à jour de Flash Player en vue de prendre en charge Adobe Access

Flash Player 10.1 et les versions ultérieures

Important : Flash Player 11.5 et les versions ultérieures intègrent le module Adobe Access ; l'étape de mise à jour (l'appel de `SystemUpdater.update(SystemUpdaterType.DRM)`) est donc inutile. Les navigateurs et plates-formes suivants sont concernés :

- Contrôle ActiveX Flash Player 11.5, pour toutes les plates-formes à l'exception d'Internet Explorer sous Windows 8
- Plug-in Flash Player 11.5, pour tous les navigateurs
- Adobe AIR (version de bureau et version mobile)

L'étape de mise à jour est *toujours requise* dans les cas suivants :

- Internet Explorer sous Windows 8
- Flash Player 11.4 et versions antérieures, à l'exception de Google Chrome 22 et des versions ultérieures (toutes les plates-formes) ou de Google Chrome 21 et des versions ultérieures (Windows)

Remarque : *vous pouvez toujours appeler `SystemUpdater.update(SystemUpdaterType.DRM)` sur un système doté de Flash Player 11.5 ou d'une version ultérieure, mais rien n'est téléchargé.*

Flash Player doit disposer du module Adobe Access pour prendre en charge ses fonctions. Lorsque Flash Player tente de lire un contenu protégé, le moteur d'exécution indique si le module ou une nouvelle version de Flash Player doit être téléchargé. Flash Player permet ainsi aux développeurs SWF de ne pas faire de mise à jour, le cas échéant.

En règle générale, pour lire du contenu protégé, les développeurs SWF mettent à jour le module ou le lecteur Adobe Access requis. Pour effectuer la mise à jour, vous disposez de l'API SystemUpdater, qui permet d'obtenir la version la plus récente du module Adobe Access ou de Flash Player.

L'API SystemUpdater n'autorise qu'une mise à jour à la fois. Le code d'erreur 2202 indique qu'une mise à jour est en cours d'exécution dans l'occurrence active du moteur d'exécution ou une autre occurrence. Ainsi, s'il se produit une mise à jour d'une occurrence de Flash Player dans Internet Explorer, il est impossible d'effectuer une mise à jour d'une occurrence de Flash Player dans Firefox.

Seules les plates-formes de bureau prennent en charge l'API SystemUpdater.

Remarque : *pour les versions de Flash Player antérieures à 10.1, utilisez le mécanisme de mise à jour pris en charge par les versions antérieures du lecteur (téléchargement et installation manuels à partir du site www.adobe.com ou d'ExpressInstall). Le programme d'installation d'AIR gère par ailleurs les mises à jour requises d'Adobe Access, mais non l'API SystemUpdater.*

Ecoute d'un événement de mise à jour

Flash Player 10.1 et les versions ultérieures

Lorsqu'une mise à jour du module Adobe Access est nécessaire, l'objet NetStream distribue un événement NetStatusEvent dont la valeur de code correspond à `DRM.UpdateNeeded`. Cette valeur indique que l'objet NetStream ne peut pas lire le flux protégé par le biais de tout module Adobe Access actuellement installé. Ecoutez cet événement et appelez le code suivant :

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

Ce code met à jour le module Adobe Access installé dans le lecteur. L'accord de l'utilisateur pour cette mise à jour du module n'est pas obligatoire.

Si le module Adobe Access est introuvable, une erreur est renvoyée. Voir l'étape 3 de « Flux de travail détaillé des API à la page 547 ».

Remarque : *si play() est appelé sur un flux chiffré dans des lecteurs antérieurs à 10.1, un événement NetStatusEvent dont la valeur de code correspond à NetStream.Play.StreamNotFound est distribué. Pour les lecteurs antérieurs, faites appel au mécanisme de mise à jour pris en charge (téléchargement et installation manuels à partir du site www.adobe.com ou d'ExpressInstall).*

S'il est nécessaire de mettre à jour le lecteur en tant que tel, l'objet SystemUpdater distribue un événement StatusEvent dont la valeur de code correspond à `DRM.UpdateNeededButIncompatible`. Pour mettre à jour le lecteur, l'accord de l'utilisateur est obligatoire. Intégrez à l'application une interface permettant à l'utilisateur d'accepter et de démarrer la mise à jour du lecteur. Ecoutez l'événement StatusEvent et appelez le code suivant :

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

Ce code démarre la mise à jour du lecteur.

D'autres événements associés à la classe SystemUpdater sont passés en revue dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Une fois la mise à jour du lecteur terminée, l'utilisateur est renvoyé à la page initiale. Le module Adobe Access est téléchargé et la lecture du flux peut démarrer.

Licences hors bande

Flash Player 11 et les versions ultérieures, Adobe AIR 3.0 et les versions ultérieures

Il est possible d'obtenir les licences hors bande (c'est-à-dire sans se connecter au serveur de licences Adobe Access) en enregistrant le voucher (la licence) sur le disque et dans la mémoire à l'aide de la méthode `storeVoucher`.

Pour lire une vidéo chiffrée dans Flash Player et AIR, le moteur d'exécution respectif doit obtenir le voucher DRM correspondant à cette vidéo. Le voucher DRM contient la clé de chiffrement de la vidéo et est généré par le serveur de licences Adobe Access que le client a déployé.

Le moteur d'exécution de Flash Player/AIR obtient généralement ce voucher en envoyant une demande de voucher au serveur de licences Adobe Access indiqué dans les métadonnées DRM de la vidéo (classe `DRMContentData`). L'application Flash/AIR peut déclencher cette demande de licence en appelant la méthode `DRMManager.loadVoucher()`. Le moteur d'exécution de Flash Player ou d'AIR peut par ailleurs solliciter automatiquement une licence au début de la lecture de la vidéo chiffrée si aucune licence ne correspond au contenu sur le disque ou dans la mémoire. Dans tous les cas, la communication avec le serveur de licences Adobe Access a une incidence sur les performances de l'application Flash/AIR.

`DRMManager.storeVoucher()` permet à l'application Flash/AIR d'envoyer les vouchers DRM obtenus hors bande au moteur d'exécution de Flash Player ou d'AIR. Le moteur d'exécution peut alors ignorer le processus de demande de licence et utiliser les vouchers transmis pour lire les vidéos chiffrées. Il est toujours nécessaire de générer le voucher DRM via le serveur de licences Adobe Access avant de pouvoir l'obtenir hors bande. Vous avez néanmoins la possibilité d'héberger les vouchers sur un serveur HTTP plutôt que sur un serveur de licences Adobe Access public.

`DRMManager.storeVoucher()` prend également en charge le partage de vouchers DRM entre plusieurs périphériques. Dans Adobe Access 3.0, cette fonction est appelée « prise en charge de domaine ». Si votre déploiement prend en charge ce cas d'utilisation, vous pouvez enregistrer plusieurs machines dans un groupe de périphériques via la méthode `DRMManager.addToDeviceGroup()`. S'il existe une machine disposant d'un voucher associé au domaine pour un contenu donné, l'application AIR peut alors extraire les vouchers DRM sérialisés à l'aide de la méthode `DRMVoucher.toByteArray()`, et il est possible d'importer les vouchers sur les autres machines à l'aide de la méthode `DRMManager.storeVoucher()`.

Enregistrement de périphérique

Les vouchers DRM sont liés à la machine de l'utilisateur final. Par conséquent, les applications Flash/AIR nécessitent un ID unique pour que la machine de l'utilisateur puisse faire référence à l'objet de voucher DRM sérialisé approprié. Le scénario suivant décrit la procédure d'enregistrement d'un périphérique :

Vous devez au préalable effectuer les tâches suivantes :

- Configurez le kit SDK du serveur Adobe Access.
- Configurez un serveur HTTP en vue d'obtenir des licences pré-générées.
- Créez une application Flash afin d'accéder au contenu protégé.

La phase d'enregistrement du périphérique implique les actions suivantes :

1. L'application Flash crée un ID généré de façon aléatoire.
2. L'application Flash invoque la méthode `DRMManager.authenticate()`. L'application doit inclure l'ID généré de façon aléatoire à la demande d'authentification, par exemple inclure l'ID dans le champ Nom d'utilisateur.
3. L'action mentionnée à l'étape 2 pousse Adobe Access à envoyer une demande d'authentification au serveur du client. Cette demande inclut le certificat du périphérique.
   a. Le serveur extrait le certificat du périphérique et l'ID généré, puis les enregistre.
   b. Le sous-système du client pré-génère les licences correspondant à ce certificat de périphérique, les enregistre et en autorise l'accès en les associant à l'ID généré.
4. Le serveur répond à la demande via un message « success ».
5. L'application Flash enregistre l'ID généré localement dans un objet partagé local (LSO).

Après l'enregistrement du périphérique, l'application Flash utilise l'ID généré de la même manière qu'elle aurait utilisé l'ID de périphérique dans le modèle précédent :

1. L'application Flash tente de localiser l'ID généré dans l'objet LSO.
2. Si l'ID généré est détecté, l'application Flash l'utilise lors du téléchargement des licences pré-générées. L'application Flash envoie les licences au client Adobe Access à l'aide de la méthode `DRMManager.storeVoucher()`.
3. Si l'ID généré est introuvable, l'application Flash suit la procédure d'enregistrement de périphérique.

Rétablissement des paramètres par défaut

Lorsque l'utilisateur du périphérique invoque l'option de rétablissement des paramètres par défaut, le certificat du périphérique est supprimé. Pour poursuivre la lecture du contenu protégé, l'application Flash doit répéter la procédure d'enregistrement de périphérique. Si l'application Flash envoie une licence pré-chargée obsolète, le client Adobe Access la refuse, car la licence a été chiffrée pour un ancien ID de périphérique.

Prise en charge de domaine

Flash Player 11 et les versions ultérieures, Adobe AIR 3.0 et les versions ultérieures

Si les métadonnées du contenu indiquent que l'enregistrement du domaine est requis, l'application AIR peut invoquer une API en vue de se joindre à un groupe de périphériques. Cette action déclenche une demande d'enregistrement à envoyer au serveur de domaine. Dès qu'une licence est délivrée à un groupe de périphériques, il est impossible de l'exporter et de la partager avec d'autres périphériques ayant rejoint le groupe.

Les informations concernant le groupe de périphériques sont alors utilisées dans l'objet `VoucherAccessInfo` du `DRMContentData` afin de présenter les informations requises pour récupérer et consommer un voucher.

Lecture de contenu chiffré à l'aide de la prise en charge de domaine

Pour lire du contenu chiffré avec Adobe Access, procédez comme suit :

1. A l'aide de `VoucherAccessInfo.deviceGroup`, vérifiez si l'enregistrement du groupe de périphériques est requis.
2. Si l'authentification est requise :
   a. Utilisez la propriété `DeviceGroupInfo.authenticationMethod` pour savoir si l'authentification est requise.
   b. Si l'authentification est requise, authentifiez l'utilisateur en appliquant l'UNE des procédures suivantes :
      - Obtenez le nom d'utilisateur et le mot de passe de l'utilisateur. Appelez `DRMManager.authenticate(deviceGroup.serverURL, deviceGroup.domain, nom d'utilisateur, mot de passe)`.
      - Obtenez un jeton d'authentification mis en cache/pré-généré et appelez `DRMManager.setAuthenticationToken()`.
   c. Appelez `DRMManager.addToDeviceGroup()`.
3. Obtenez le voucher correspondant au contenu en appliquant l'une des procédures suivantes :
   a. Utilisez la méthode `DRMManager.loadVoucher()`.
   b. Obtenez le voucher à partir d'un autre périphérique enregistré dans le même groupe de périphériques. Fournissez le voucher au `DRMManager` via la méthode `DRMManager.storeVoucher()`.
4. Lisez le contenu chiffré à l'aide de la méthode `NetStream.play()`.

Pour exporter la licence correspondant au contenu, n'importe quel périphérique peut fournir les octets bruts de la licence à l'aide de la méthode `DRMVoucher.toByteArray()` une fois la licence obtenue auprès du serveur de licences Adobe Access. Le fournisseur de contenu limite généralement le nombre de périphériques dans un groupe de périphériques. Si vous atteignez cette limite, il est possible que vous deviez appeler la méthode `DRMManager.removeFromDeviceGroup()` sur un périphérique non utilisé avant d'enregistrer le périphérique actuel.

Aperçu de la licence

L'application Flash peut envoyer une demande d'aperçu de licence, c'est-à-dire qu'elle peut lancer une opération d'aperçu avant de prier l'utilisateur d'acheter le contenu en vue de déterminer si la machine de ce dernier répond aux critères de lecture requis. L'aperçu de licence signifie que le client est en mesure d'afficher un aperçu de la licence (pour connaître les droits octroyés par la licence), mais qu'il ne peut pas afficher un aperçu du contenu (c.-à-d. afficher une petite partie du contenu avant de décider de l'acheter). Certains des paramètres uniques à chaque machine sont les suivants : sorties disponibles et statuts de protection associés, version du moteur d'exécution/client DRM disponible, niveau de sécurité du client DRM, etc. Le mode d'aperçu de licence permet au moteur d'exécution/client DRM de tester la logique métier du serveur de licences et de fournir des informations à l'utilisateur afin que ce dernier puisse prendre une décision. Le client peut donc connaître l'aspect d'une licence valide, sans pour autant recevoir la clé permettant de déchiffrer le contenu. La prise en charge du mode d'aperçu de licence est facultative et uniquement nécessaire si vous implémentez une application personnalisée qui a recours à cette fonctionnalité.

Diffusion de contenu

Adobe Access ne s'attache pas au mécanisme de diffusion de contenu, car Flash Player abstrait la couche de mise en réseau et se contente de diffuser le contenu protégé au sous-système de Adobe Access. Il est donc possible de diffuser le contenu via HTTP, diffusion en continu dynamique HTTP, RTMP ou RTMPE.

Il est toutefois possible que des problèmes se produisent en raison de la nécessité d'obtenir les métadonnées du contenu protégé (en général sous forme de fichier .metadata) avant qu'Adobe Access ne puisse acquérir une licence pour déchiffrer le contenu. Plus spécifiquement, avec le protocole RTMP/RTMPE, seules les données FLV et F4V peuvent être diffusées au client via Flash Media Server (FMS). Le client doit par conséquent utiliser d'autres moyens de récupérer l'objet BLOB de métadonnées. Pour résoudre ce problème, il est possible d'héberger les métadonnées sur un serveur Web HTTP et d'implémenter le lecteur vidéo client en vue de récupérer les métadonnées correspondantes, en fonction du contenu en cours de lecture.

```
private function getMetadata():void{

    extrapolated-path-to-metadata = "http://metadatas.mywebserver.com/" + videoname;
    var urlRequest : URLRequest = new URLRequest(extrapolated-path-to-the-metadata + ".metadata");
    var urlStream : URLStream = new URLStream();
    urlStream.addEventListener(Event.COMPLETE, handleMetadata);
    urlStream.addEventListener(IOErrorEvent.NETWORK_ERROR, handleIOError);
    urlStream.addEventListener(IOErrorEvent.IO_ERROR, handleIOError);
    urlStream.addEventListener(IOErrorEvent.VERIFY_ERROR, handleIOError);
    try{
        urlStream.load(urlRequest);
     }catch(se:SecurityError){
        videoLog.text += se.toString() + "\n";
     }catch(e:Error){
        videoLog.text += e.toString() + "\n";
     }
}
```

Open Source Media Framework

Open Source Media Framework (OSMF) est une structure d'application basée sur ActionScript qui laisse une entière liberté et un contrôle absolu à l'utilisateur lors de la création d'expériences multimédias enrichies. Pour plus d'informations sur OSMF, consulter la page consacrée aux développeurs d'OSMF.

Procédure de lecture de contenu protégé

1 Créez une occurrence de `MediaPlayer`.

```
player = new MediaPlayer();
```

2 Enregistrez l'événement `MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE` dans le lecteur. Cet événement est distribué si le contenu est protégé par DRM.

```
player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE,
onDRMCapabilityChange);
```

3 Dans le gestionnaire d'événement, obtenez l'occurrence de `DRMTrait`. `DRMTrait` est l'interface par le biais de laquelle vous appelez les méthodes associées au client DRM, telles que `authenticate()`. Lors du chargement d'un contenu protégé par DRM, OSMF procède à la gestion de droits multimédias en validant les actions, puis distribue des événements d'état. Ajoutez un gestionnaire d'événement `DRMEvent.DRM_STATE_CHANGE` à l'interface `DRMTrait`.

```
private function onDRMCapabilityChange
    (event :MediaPlayerCapabilityChangeEvent) :void
        {
            if (event.type == MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE
                && event.enabled)
            {
                drmTrait = player.media.getTrait(MediaTraitType.DRM) as DRMTrait;
                drmTrait.addEventListener
                    (DRMEvent.DRM_STATE_CHANGE, onDRMStateChange);
            }
        }
```

4 Gérez les événements DRM dans la méthode `onDRMStateChange()`.

```
        private function onDRMStateChange(event :DRMEvent) :void
        {
            trace ( "DRMState: ",event.drmState);
            switch(event.drmState)
            {
                case DRMState.AUTHENTICATION_NEEDED:
                    // Identity-based content
                    var authPopup :AuthWindow = AuthWindow.create(_parentWin);
                    authPopup.serverURL = event.serverURL;
                    authPopup.addEventListener("dismiss", function () :void {
                        trace ("Authentication dismissed");
                        if(_drmTrait != null)
                        {
                            //Ignore authentication. Just
                            //try to acquire a license.
                            _drmTrait.authenticate(null, null);
                        }
                    });
                    authPopup.addEventListener("authenticate",
                                    function (event :AuthWindowEvent) :void {
```

```
                if(_drmTrait != null)
                {
                    _drmTrait.authenticate(event.username, event.password);
                }
            });
            authPopup.show();
            break;
        case DRMState.AUTHENTICATING:
            //Display any authentication message.
            trace("Authenticating...");
            break;
        case DRMState.AUTHENTICATION_COMPLETE:
            // Start to retrieve voucher and playback.
            // You can display the voucher information at this point.
            if(event.token)
            // You just received the authentication token.
            {
                trace("Authentication success. Token: \n", event.token);
            }
            else
            // You have got the voucher.
            {
                trace("DRM License:");
                trace("Playback window period: ",
                    !isNaN(event.period) ? event.period == 0 ?
                    "<unlimited>" : event.period : "<none>");
                trace("Playback window end date: ",
                    event.endDate != null ? event.endDate : "<none>");
                trace("Playback window start date: ",
                    event.startDate != null ? event.startDate : "<none>");
            }
            break;
        case DRMState.AUTHENTICATION_ERROR:
            trace ("DRM Error:", event.mediaError.errorID +
                "[" + DRMErrorEventRef.getDRMErrorMnemonic
                (event.mediaError.errorID) + "]");
            //Stop everything.
            player.media = null;
            break;
        case DRMState.DRM_SYSTEM_UPDATING:
            Logger.log("Downloading DRM module...");
            break;
        case DRMState.UNINITIALIZED:
            break;
    }
}
```

Chapitre 28 : Ajout d'un contenu PDF dans AIR

Adobe AIR 1.0 et les versions ultérieures

Les applications qui s'exécutent sous Adobe® AIR® peuvent non seulement créer un contenu SWF ou HTML, mais également un contenu PDF. Les applications AIR créent un contenu PDF à l'aide de la classe HTMLLoader, du moteur WebKit et du module d'extension du navigateur Adobe® Reader®. Dans une application AIR, le contenu PDF peut s'étendre sur toute la hauteur et toute la largeur de votre application ou bien sur une partie de l'interface. Les contrôles du module d'extension du navigateur Adobe Reader affichent les fichiers PDF dans une application AIR. Les modifications apportées à l'interface de la barre d'outils de Reader (les contrôles de position, d'ancrage et de visibilité, par exemple) sont réactivées lors de l'affichage ultérieur de fichiers PDF dans les applications AIR et le navigateur.

Important : *pour afficher un contenu PDF dans AIR, l'utilisateur doit avoir installé Adobe Reader ou Adobe® Acrobat® version 8.1 (ou une version ultérieure).*

Détection des capacités PDF

Adobe AIR 1.0 et les versions ultérieures

Si l'utilisateur ne dispose pas d'Adobe Reader ou d'Adobe Acrobat 8.1 ou ultérieur, le contenu PDF ne s'affiche pas dans une application AIR. Pour vous rendre compte si un utilisateur est en mesure de produire un contenu PDF, il faut d'abord vérifier la propriété `HTMLLoader.pdfCapability`. Elle est définie sur l'une des constantes suivantes de la classe HTMLPDFCapability :

Constante	Description
HTMLPDFCapability.STATUS_OK	Une version appropriée (8.1 ou plus récente) d'Adobe Reader a été décelée et le contenu PDF peut être chargé dans un objet HTMLLoader.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	Aucune version d'Adobe Reader n'a été décelée. Un objet HTMLLoader n'est pas en mesure d'afficher le contenu PDF.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Adobe Reader a bien été décelé, mais la version est périmée. Un objet HTMLLoader n'est pas en mesure d'afficher le contenu PDF.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Une version appropriée (8.1 ou plus récente) d'Adobe Reader a été détectée mais celle qui est configurée pour traiter le contenu PDF est plus ancienne que Reader 8.1. Un objet HTMLLoader n'est pas en mesure d'afficher le contenu PDF.

Sous Windows, si Adobe Acrobat ou Adobe Reader version 7.x (ou une version ultérieure) s'exécute sur le système de l'utilisateur, c'est cette version-là qui est utilisée, même si une version plus récente qui prend en charge les contenus PDF est installée. Dans ce cas, si la valeur de la propriété `pdfCapability` est `HTMLPDFCapability.STATUS_OK`, lorsqu'une application AIR tente de charger un contenu PDF, la version la plus ancienne d'Acrobat ou de Reader affiche un message d'avertissement et aucune exception n'est renvoyée dans l'application AIR. Si cette situation est susceptible de se produire chez vos utilisateurs finaux, pensez à leur fournir des instructions pour qu'ils quittent Acrobat tandis qu'ils exécutent leur application. Vous pourriez afficher ces instructions si le contenu PDF ne se charge pas dans un délai raisonnable.

Sous Linux, AIR recherche Adobe Reader dans le chemin exporté par l'utilisateur (s'il contient la commande acroread) et dans le répertoire /opt/Adobe/Reader.

Le code suivant détecte si un utilisateur est en mesure d'afficher les contenus PDF dans une application AIR. Si tel n'est pas le cas, le code recherche le code d'erreur correspondant à l'objet d'erreur HTMLPDFCapability :

```
 if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

Chargement du contenu PDF

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez ajouter un PDF à une application AIR en créant une occurrence de HTMLLoader, en paramétrant ses dimensions et en chargeant le chemin d'un PDF.

L'exemple suivant charge un PDF à partir d'un site externe. Remplacez l'URLRequest par le chemin qui mène à un PDF externe disponible.

```
 var request:URLRequest = new URLRequest("http://www.example.com/test.pdf");
pdf = new HTMLLoader();
pdf.height = 800;
pdf.width = 600;
pdf.load(request);
container.addChild(pdf);
```

Vous pouvez également charger du contenu depuis des modèles d'URL file et des modèles d'URL spécifiques à AIR, comme app et app-storage. Par exemple, le code suivant charge le fichier test.pdf dans le sous-répertoire du PDF contenu dans le répertoire de l'application :

app:/js_api_reference.pdf

Pour plus d'informations sur les modèles d'URL d'AIR, voir « Modèles d'URI » à la page 845.

Programmation du contenu PDF

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser JavaScript pour contrôler le contenu PDF de la même façon que vous le feriez dans une page Web du navigateur.

Les extensions JavaScript incorporées dans Acrobat fournissent les fonctions suivantes, entre autres :

- Contrôle de la navigation et de l'agrandissement de la page
- Traitement des formulaires au sein du document
- Contrôle des événements multimédia

Des détails complets sur les extensions de JavaScript pour Adobe Acrobat se trouvent sur le site Adobe Acrobat Developer Connection à l'adresse http://www.adobe.com/devnet/acrobat/javascript.html.

Principes de communication HTML-PDF

Adobe AIR 1.0 et les versions ultérieures

JavaScript dans une page HTML peut envoyer un message à JavaScript dans un contenu PDF par un appel à la méthode `postMessage()` de l'objet DOM représentant le contenu PDF. Par exemple, examinez le contenu PDF intégré ci-dessous :

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

Le code JavaScript suivant dans le contenu HTML envoie un message au JavaScript du fichier PDF :

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

Le fichier PDF peut contenir JavaScript pour recevoir ce message. Vous pouvez ajouter du code JavaScript aux fichiers PDF dans certains contextes au niveau du document, du dossier, de la page, du champ ou du lot. Nous n'aborderons ici que le contexte au niveau du document, celui qui définit les scripts qui sont évalués lorsque le document PDF s'ouvre.

Un fichier PDF peut ajouter une propriété `messageHandler` à l'objet `hostContainer`. La propriété `messageHandler` est un objet qui définit les fonctions du gestionnaire pour répondre aux messages. Par exemple, le code ci-dessous définit la fonction appelée à répondre aux messages provenant du fichier PDF du conteneur hôte. Celui-ci constitue le contenu HTML qui a intégré le fichier PDF :

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

Le code JavaScript de la page HTML peut appeler la méthode `postMessage()` de l'objet PDF contenu dans la page. L'appel de cette méthode envoie un message `"Hello from HTML"` au JavaScript, au niveau du document, dans le fichier PDF :

```
<html>
    <head>
    <title>PDF Test</title>
    <script>
        function init()
        {
            pdfObject = document.getElementById("PDFObj");
            try {
                 pdfObject.postMessage(["alert", "Hello from HTML"]);
            }
            catch (e)
            {
                alert( "Error: \n name = " + e.name + "\n message = " + e.message );
            }
        }
    </script>
    </head>
    <body onload='init()'>
        <object
            id="PDFObj"
            data="test.pdf"
            type="application/pdf"
            width="100%" height="100%"/>
    </body>
</html>
```

Pour consulter un exemple plus complexe, ainsi que des informations sur l'utilisation d'Acrobat 8 pour ajouter du code JavaScript à un fichier PDF, voir Programmation croisée du contenu PDF dans Adobe AIR.

Programmation du contenu PDF depuis ActionScript

Adobe AIR 1.0 et les versions ultérieures

Le code ActionScript (dans un contenu SWF) ne peut pas communiquer directement avec JavaScript dans un contenu PDF. Toutefois, ActionScript peut communiquer avec le JavaScript de la page HTML chargée dans un objet HTMLLoader qui charge le contenu PDF. Ce code JavaScript peut communiquer avec le JavaScript dans le fichier PDF chargé. Pour plus d'informations, voir « Programmation HTML et JavaScript dans AIR » à la page 1019.

Limites connues pour du contenu PDF dans AIR

Adobe AIR 1.0 et les versions ultérieures

Le contenu PDF dans Adobe AIR a les limites suivantes :

- Le contenu PDF ne peut pas s'afficher dans une fenêtre (un objet NativeWindow) qui est transparente et où la propriété `transparent` est définie sur `true`.

- L'ordre d'affichage d'un fichier PDF s'effectue différemment que les autres objets à afficher dans une application AIR. Bien que le contenu PDF se découpe correctement en respectant l'ordre d'affichage HTML, il s'installera toujours par-dessus le contenu dans l'ordre d'affichage de l'application AIR.
- Si certaines propriétés visuelles de l'objet HTMLLoader contenant un document PDF sont modifiées, le document PDF devient invisible. Ces propriétés comprennent `filters`, `alpha`, `rotation` et `scaling`. Si vous modifiez ces propriétés, le contenu PDF devient invisible jusqu'à ce que vous les réinitialisez. Le contenu PDF est également invisible si vous modifiez ces propriétés dans les conteneurs d'objets d'affichage qui contiennent l'objet HTMLLoader.
- Le contenu du PDF n'est visible que lorsque la propriété `scaleMode` de l'objet Stage de l'objet NativeWindow renfermant le contenu PDF est définie sur `StageScaleMode.NO_SCALE`. Lorsqu'elle est définie sur une autre valeur, le contenu PDF n'est pas visible.
- Un clic sur des liens pointant vers le contenu au sein du fichier PDF met à jour la position de défilement du contenu PDF. Un clic sur des liens pointant vers le contenu hors d'un fichier PDF réoriente l'objet HTMLLoader qui contient le PDF, même si la cible d'un lien est une nouvelle fenêtre.
- Les flux de commentaires PDF ne fonctionnent pas dans AIR.

Chapitre 29 : Principes de base de l'interaction utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Votre application prend en charge l'interactivité en utilisant ActionScript 3.0 pour réagir à une action utilisateur. Cette section suppose que vous maîtrisez le modèle d'événement d'ActionScript 3.0. Pour plus d'informations, voir « Gestion des événements » à la page 129.

Capture des entrées utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'interaction utilisateur, au moyen du clavier, de la souris, de la caméra ou d'une combinaison de ces périphériques, est à la base de l'interactivité. Dans ActionScript 3.0, l'identification et la réponse à l'interaction utilisateur impliquent principalement l'écoute d'événements.

La classe InteractiveObject, une sous-classe de la classe DisplayObject, fournit la structure d'événements courante et la fonctionnalité nécessaire à la gestion de l'interaction utilisateur. Vous ne créez pas directement une occurrence de la classe InteractiveObject. Affichez plutôt des objets tels SimpleButton, Sprite, TextField et ainsi des composants divers de l'outil de programmation Flash et de Flex héritent leur modèle d'interaction utilisateur à partir de cette classe. Ils partagent alors une structure commune. Cela signifie que les techniques que vous apprenez et le code que vous écrivez pour gérer l'interaction utilisateur dans un objet dérivé de InteractiveObject sont applicables à tous les autres.

Concepts importants et terminologie

Il est important que vous vous familiarisiez avec les termes d'interaction utilisateur suivants avant de poursuivre :

Code de caractère Code numérique représentant un caractère dans le jeu de caractères actuel (associé à une touche tapée sur le clavier). Par exemple, D et d ont des codes de caractères différents même si elles sont créées par la même touche sur un clavier français.

Menu contextuel Menu qui apparaît lorsqu'un utilisateur clique avec le bouton droit de la souris ou utilise une combinaison clavier-souris particulière. Les commandes de menu contextuel s'appliquent généralement directement à l'élément sur lequel vous avez cliqué. Par exemple, un menu contextuel pour une image peut contenir une commande pour afficher l'image dans une fenêtre séparée et une commande pour la télécharger.

Cible d'action Indication qu'un élément sélectionné est actif et qu'il est la cible d'une interaction clavier ou souris.

Code de touche Code numérique correspondant à une touche physique du clavier.

Voir aussi

InteractiveObject

Keyboard

Mouse

ContextMenu

Gestion de la cible d'action

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet interactif peut recevoir le focus, soit par programmation, soit par le biais d'une action utilisateur. En outre, si la propriété `tabEnabled` a la valeur `true`, l'utilisateur peut transmettre le focus d'un objet à un autre en appuyant sur la touche Tabulation. La valeur `tabEnabled` est `false` par défaut, excepté dans les cas suivants :

- Pour un objet SimpleButton, la valeur est `true`.
- Pour un champ de texte d'entrée, la valeur est `true`.
- Pour un objet Sprite ou MovieClip dont `buttonMode` a la valeur `true`, la valeur est `true`.

Dans chacune de ces situations, vous pouvez ajouter un écouteur pour `FocusEvent.FOCUS_IN` ou `FocusEvent.FOCUS_OUT` pour étendre les comportements possibles lors du changement de focus. Si cette technique est pratique pour les champs texte et les formulaires, vous pouvez aussi l'utiliser avec les sprites, les clips et tout autre objet qui hérite de la classe InteractiveObject. L'exemple suivant montre comment activer le changement de focus avec la touche de tabulation et comment répondre à l'événement focus qui en découle. Dans ce cas, chaque carré change de couleur lorsqu'il reçoit le focus.

Remarque : *Flash Professional utilise des raccourcis clavier pour gérer le focus. Par conséquent, pour simuler correctement la gestion du focus, les fichiers SWF doivent être testés dans un navigateur ou dans AIR plutôt que dans Flash.*

```
var rows:uint = 10;
var cols:uint = 10;
var rowSpacing:uint = 25;
var colSpacing:uint = 25;
var i:uint;
var j:uint;
for (i = 0; i < rows; i++)
{
    for (j = 0; j < cols; j++)
    {
        createSquare(j * colSpacing, i * rowSpacing, (i * cols) + j);
    }
}

function createSquare(startX:Number, startY:Number, tabNumber:uint):void
{
    var square:Sprite = new Sprite();
    square.graphics.beginFill(0x000000);
    square.graphics.drawRect(0, 0, colSpacing, rowSpacing);
    square.graphics.endFill();
    square.x = startX;
```

```
    square.y = startY;
    square.tabEnabled = true;
    square.tabIndex = tabNumber;
    square.addEventListener(FocusEvent.FOCUS_IN, changeColor);
    addChild(square);
}
function changeColor(event:FocusEvent):void
{
    event.target.transform.colorTransform = getRandomColor();
}
function getRandomColor():ColorTransform
{
    // Generate random values for the red, green, and blue color channels.
    var red:Number = (Math.random() * 512) - 255;
    var green:Number = (Math.random() * 512) - 255;
    var blue:Number = (Math.random() * 512) - 255;

    // Create and return a ColorTransform object with the random colors.
    return new ColorTransform(1, 1, 1, 1, red, green, blue, 0);
}
```

Découverte des types de saisie

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Les versions de Flash Player 10.1 et d'Adobe AIR 2 ont introduit la capacité de tester l'environnement d'exécution pour vérifier s'il prend en charge des types de saisie donnés. ActionScript permet de tester si le périphérique sur lequel le moteur d'exécution est actuellement déployé :

- prend en charge la saisie par stylet ou à l'aide du doigt (ou bien aucune saisie tactile) ;
- propose à l'utilisateur un clavier virtuel ou physique (ou bien aucun clavier) ;
- affiche un curseur (si tel n'est pas le cas, les fonctions qui s'appuient sur le survol d'un objet par le curseur ne fonctionnent pas).

Les API de découverte de la saisie d'ActionScript sont les suivantes :

- Propriété flash.system.Capabilities.touchscreenType : valeur fournie à l'exécution, qui indique le type de saisie pris en charge dans l'environnement actuel.
- Classe flash.system.TouchscreenType : classe de constantes de valeur d'énumération pour la propriété Capabilities.touchscreenType.
- Propriété flash.ui.Mouse.supportsCursor : valeur entrée à l'exécution qui indique si un curseur permanent est disponible ou non.
- Propriété flash.ui.Keyboard.physicalKeyboardType : valeur entrée à l'exécution qui indique si un clavier physique complet ou un pavé numérique uniquement est disponible, ou bien aucun clavier.
- Classe flash.ui.KeyboardType : classe de constantes de valeurs d'énumération associée à la propriété flash.ui.Keyboard.physicalKeyboardType.
- Propriété flash.ui.Keyboard.hasVirtualKeyboard : valeur entrée à l'exécution qui indique si l'utilisateur dispose d'un clavier virtuel (au lieu d'un clavier physique ou en plus de ce dernier).

Les API de découverte de saisie permettent d'exploiter les capacités du périphérique de l'utilisateur ou proposent une autre solution lorsque ces capacités n'existent pas. Ces API s'avèrent particulièrement utiles pour développer des applications tactiles et mobiles. Ainsi, si l'interface d'un périphérique mobile possède des boutons de petite taille adaptés à un stylet, vous pouvez proposer une autre interface dotée de boutons de taille supérieure que l'utilisateur peut toucher du doigt. Le code suivant se rapporte à une application dont la fonction createStylusUI() affecte un jeu d'éléments d'interface utilisateur adaptés à l'interaction avec un stylet. Une autre fonction, appelée createTouchUI(), affecte un autre jeu d'éléments d'interface utilisateur adaptés à l'interaction tactile :

```
if(Capabilities.touchscreenType == TouchscreenType.STYLUS ){
    //Construct the user interface using small buttons for a stylus
    //and allow more screen space for other visual content
    createStylusUI();
} else if(Capabilities.touchscreenType = TouchscreenType.FINGER){
    //Construct the user interface using larger buttons
    //to capture a larger point of contact with the device
    createTouchUI();
}
```

Lorsque vous développez des applications destinées à des environnements de saisie différents, tenez compte du tableau de compatibilité suivant :

Environnement	supportsCursor	touchscreenType == FINGER	touchscreenType == STYLUS	touchscreenType == NONE
Bureau traditionnel	true	false	false	true
Périphériques équipés d'un écran tactile à technologie capacitive (tablettes, organisateurs personnels et téléphones qui détectent une action tactile humaine subtile, tel l'iPhone d'Apple ou le Palm Pre)	false	true	false	false
Périphériques équipés d'un écran tactile à technologie résistive (tablettes, organisateurs personnels et téléphones qui détectent un contact précis à pression élevée, tel le HTC Fuze)	false	false	true	false
Périphériques sans écran tactile (téléphones et périphériques qui exécutent des applications, mais dont l'écran ne détecte pas le contact)	false	false	false	true

Remarque : *diverses plates-formes gèrent un grand nombre de combinaisons de types de saisie. Référez-vous à ce tableau à titre de référence.*

Chapitre 30 : Saisie au clavier

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Votre application peut capturer la saisie au clavier et réagir à celle-ci et peut manipuler un IME pour permettre aux utilisateurs de taper des caractères non ASCII dans les langues codées sur plusieurs octets. Cette section suppose que vous maîtrisez le modèle d'événement d'ActionScript 3.0. Pour plus d'informations, voir « Gestion des événements » à la page 129.

Pour plus d'informations sur l'identification du clavier pris en charge (physique, virtuel, alphanumérique, numérique à 12 touches, etc.) à l'exécution, voir « Découverte des types de saisie » à la page 576.

Grâce à un IME (Input Method Editor), l'utilisateur est en mesure de taper des caractères et des symboles complexes sur un clavier standard. L'utilisation des classes IME permet à l'utilisateur de bénéficier de l'IME système dans une application.

Voir aussi

flash.events.KeyboardEvent

flash.system.IME

Capture de la saisie au clavier

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les objets d'affichage qui héritent de leur modèle d'interaction de la classe InteractiveObject peuvent répondre à des événements de clavier à l'aide d'écouteurs d'événement. Par exemple, vous pouvez placer un écouteur d'événement sur la scène pour écouter et répondre à une saisie au clavier. Dans le code suivant, un écouteur d'événement capture une saisie au clavier et le nom et les propriétés de code de la touche sont affichés :

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " +
event.charCode + ")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Certaines touches (Ctrl, par exemple) génèrent des événements, même si elles n'ont pas de représentation de glyphes.

Dans l'exemple de code précédent, l'écouteur d'événement de clavier a capturé une saisie au clavier pour la scène entière. Vous pouvez également écrire un écouteur d'événement pour un objet d'affichage spécifique sur la scène ; cet écouteur d'événement est déclenché lorsque l'objet a le focus.

Dans l'exemple suivant, les frappes de touches apparaissent dans le panneau Sortie uniquement lorsque l'utilisateur tape dans l'occurrence de TextField. S'il maintient la touche Maj enfoncée, la couleur du contour du TextField devient temporairement rouge.

Ce code suppose qu'il existe une occurrence de TextField appelée `tf` sur la scène.

```
tf.border = true;
tf.type = "input";
tf.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);
tf.addEventListener(KeyboardEvent.KEY_UP,reportKeyUp);

function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (key code: " +
event.keyCode + " character code: " + event.charCode + ")");
    if (event.keyCode == Keyboard.SHIFT) tf.borderColor = 0xFF0000;
}

function reportKeyUp(event:KeyboardEvent):void
{
    trace("Key Released: " + String.fromCharCode(event.charCode) + " (key code: " +
event.keyCode + " character code: " + event.charCode + ")");
    if (event.keyCode == Keyboard.SHIFT)
    {
        tf.borderColor = 0x000000;
    }
}
```

La classe TextField signale également un événement `textInput` que vous pouvez écouter lorsqu'un utilisateur saisit du texte. Pour plus d'informations, voir « Capture du texte saisi par l'utilisateur » à la page 391.

Remarque : *dans le moteur d'exécution d'AIR, il est possible d'annuler un événement de clavier. En revanche, dans le moteur d'exécution de Flash Player, un événement de clavier ne peut pas être annulé.*

Codes de touches et de caractères

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les propriétés `keyCode` et `charCode` d'un événement clavier permettent de déterminer la touche utilisée et de déclencher d'autres actions. La propriété`keyCode` est une valeur numérique qui correspond à la valeur de la touche sur le clavier. La propriété `charCode` est la valeur numérique de cette touche dans le jeu de caractères actuel (le jeu de caractères par défaut est UTF-8, qui prend en charge ASCII).

La différence principale entre le code de touche et les valeurs de caractères est la suivante : la valeur du code de touche représente une touche déterminée du clavier (la touche 1 sur le pavé numérique est différente du 1 sur le clavier central, mais cette dernière permet à la fois de générer 1 et &) alors que la valeur du caractère représente un caractère particulier (les caractères R et r sont différents).

Remarque : *pour plus d'informations sur le mappage entre les touches et le code de caractère ASCII correspondant, voir la classe flash.ui.Keyboard dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.*

Les mappages entre les touches et leurs codes de touches dépendent du périphérique et du système d'exploitation. C'est pourquoi vous ne devez pas utiliser de mappages de touches pour déclencher des actions. A la place, utilisez les valeurs de constante prédéfinies fournies par la classe Keyboard pour référencer les propriétés `keyCode` appropriées. Par exemple, au lieu d'utiliser le mappage de touche pour la touche Maj, utilisez la constante `Keyboard.SHIFT` (comme indiqué dans l'exemple de code précédent).

Priorité des événements KeyboardEvent

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme avec d'autres événements, la séquence d'événement de clavier est déterminée par la hiérarchie d'objet d'affichage et non par l'ordre dans lequel les méthodes `addEventListener()` sont affectées dans le code.

Supposons par exemple que vous placiez un champ de texte `tf` au sein d'un clip nommé `container` et que vous ajoutiez un écouteur d'événement pour l'événement de clavier à chaque occurrence, comme indiqué dans l'exemple suivant :

```
container.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);
container.tf.border = true;
container.tf.type = "input";
container.tf.addEventListener(KeyboardEvent.KEY_DOWN,reportKeyDown);

function reportKeyDown(event:KeyboardEvent):void
{
    trace(event.currentTarget.name + " hears key press: " + String.fromCharCode(event.charCode)
+ " (key code: " + event.keyCode + " character code: " + event.charCode + ")");
}
```

Etant donné qu'il existe un écouteur sur le champ de texte et sur son conteneur parent, la fonction `reportKeyDown()` est appelée deux fois pour chaque frappe de touche dans le TextField. Notez que pour chaque touche actionnée, le champ de texte envoie un événement avant que le clip `container` ne distribue un événement.

Le système d'exploitation et le navigateur Web traiteront les événements de clavier avant Adobe Flash Player ou AIR. Par exemple, dans Microsoft Internet Explorer, lorsque vous appuyez sur Ctrl+W, vous fermez la fenêtre du navigateur avant qu'un fichier SWF contenu ne distribue un événement de clavier.

Utilisation de la classe IME

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe IME permet de manipuler l'IME du système d'exploitation à partir de Flash Player ou Adobe AIR.

A l'aide d'ActionScript, vous pouvez déterminer les éléments suivants :

- Si un IME est installé sur l'ordinateur de l'utilisateur (`Capabilities.hasIME`)
- Si l'IME est activé ou désactivé sur l'ordinateur de l'utilisateur (`IME.enabled`)
- Le mode de conversion utilisé par l'IME actif (`IME.conversionMode`)

Vous pouvez associer un champ de saisie de texte à un contexte IME particulier. Lorsque vous passez d'un champ de saisie à un autre, vous pouvez également changer l'IME pour utiliser les caractères Hiragana (japonais), des nombres à pleine chasse, des nombres à demi-chasse, la saisie directe, etc.

Un IME permet aux utilisateurs d'entrer des caractères de texte non ASCII des langues codées sur plusieurs octets, telles que le chinois, le japonais et le coréen.

Pour plus d'informations sur les IME, voyez la documentation du système d'exploitation correspondant à la plate-forme pour laquelle vous développez l'application. Pour davantage de ressources, voir également les sites Web suivants :

- http://www.msdn.microsoft.com/goglobal/

- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

Remarque : *si aucun IME n'est actif sur l'ordinateur de l'utilisateur, tout appel aux méthodes ou propriétés IME, autres que* `Capabilities.hasIME`*, échoue. Lorsque vous activez manuellement un IME, les appels ActionScript suivants aux méthodes et aux propriétés IME fonctionnent comme prévu. Par exemple, si vous utilisez un IME japonais, vous devez l'activer avant d'appeler une méthode ou une propriété IME.*

Confirmation de l'installation et de l'activation d'un IME

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Avant d'appeler des méthodes ou propriétés IME, vous devez toujours vérifier si un IME est installé et activé sur l'ordinateur de l'utilisateur. Le code suivant montre comment vérifier que l'utilisateur dispose d'un IME installé et activé avant d'appeler une méthode :

```
if (Capabilities.hasIME)
{
    if (IME.enabled)
    {
        trace("IME is installed and enabled.");
    }
    else
    {
        trace("IME is installed but not enabled. Please enable your IME and try again.");
    }
}
else
{
    trace("IME is not installed. Please install an IME and try again.");
}
```

Le code précédent commence par vérifier si un IME est installé à l'aide la propriété `Capabilities.hasIME`. Si la valeur de la propriété est `true`, le code vérifie ensuite si l'IME est activé à l'aide de la propriété `IME.enabled`.

Identification du mode de conversion IME activé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous construisez une application multilingue, il peut être nécessaire de déterminer le mode de conversion actif dans le système d'exploitation. Le code suivant montre comment vérifier si l'utilisateur dispose d'un IME installé et, le cas échéant, quel mode de conversion IME est activé :

```
if (Capabilities.hasIME)
{
    switch (IME.conversionMode)
    {
        case IMEConversionMode.ALPHANUMERIC_FULL:
            tf.text = "Current conversion mode is alphanumeric (full-width).";
            break;
        case IMEConversionMode.ALPHANUMERIC_HALF:
            tf.text = "Current conversion mode is alphanumeric (half-width).";
            break;
        case IMEConversionMode.CHINESE:
            tf.text = "Current conversion mode is Chinese.";
            break;
        case IMEConversionMode.JAPANESE_HIRAGANA:
            tf.text = "Current conversion mode is Japananese Hiragana.";
            break;
        case IMEConversionMode.JAPANESE_KATAKANA_FULL:
            tf.text = "Current conversion mode is Japanese Katakana (full-width).";
            break;
        case IMEConversionMode.JAPANESE_KATAKANA_HALF:
            tf.text = "Current conversion mode is Japanese Katakana (half-width).";
            break;
        case IMEConversionMode.KOREAN:
            tf.text = "Current conversion mode is Korean.";
            break;
        default:
            tf.text = "Current conversion mode is " + IME.conversionMode + ".";
            break;
    }
}
else
{
    tf.text = "Please install an IME and try again.";
}
```

Le code ci-dessus commence par vérifier si l'utilisateur dispose d'un IME. Ensuite, il vérifie le mode de conversion actuellement utilisé par l'IME en comparant la propriété `IME.enabled` à chacune des constantes de la classe IMEConversionMode.

Définition du mode de conversion IME

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous modifiez le mode de conversion de l'IME de l'utilisateur, veillez à ce que le code soit enveloppé dans un bloc `try..catch`, car la définition du mode de conversion à l'aide de la propriété `conversionMode` peut donner lieu à une erreur si l'IME ne peut pas définir le mode choisi. Le code suivant illustre l'utilisation d'un bloc `try..catch` lors de la définition de la propriété`IME.conversionMode` :

```
var statusText:TextField = new TextField;
statusText.autoSize = TextFieldAutoSize.LEFT;
addChild(statusText);
if (Capabilities.hasIME)
{
    try
    {
        IME.enabled = true;
        IME.conversionMode = IMEConversionMode.KOREAN;
        statusText.text = "Conversion mode is " + IME.conversionMode + ".";
    }
    catch (error:Error)
    {
        statusText.text = "Unable to set conversion mode.\n" + error.message;
    }
}
```

Ce code commence par créer un champ de texte qui sert à afficher un message d'état à l'intention de l'utilisateur. Ensuite, si l'IME est installé, le code l'active et définit le mode de conversion coréen. Si l'ordinateur de l'utilisateur ne dispose pas d'un IME coréen, une erreur est renvoyée par Flash Player ou AIR et interceptée par le bloc `try..catch`. Le bloc `try..catch` affiche le message d'erreur dans le champ de texte créé précédemment.

Désactivation de l'IME pour certains champs de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans certains cas, il peut être nécessaire de désactiver l'IME de l'utilisateur pendant que ce dernier saisit des caractères. Par exemple, si un champ de texte accepte uniquement des caractères numériques, il peut être préférable d'éviter l'intervention de l'IME pour ne pas ralentir la saisie des données.

L'exemple suivant montre comment écouter les événements `FocusEvent.FOCUS_IN` et `FocusEvent.FOCUS_OUT` et désactiver l'IME de l'utilisateur en conséquence :

```
var phoneTxt:TextField = new TextField();
var nameTxt:TextField = new TextField();

phoneTxt.type = TextFieldType.INPUT;
phoneTxt.addEventListener(FocusEvent.FOCUS_IN, focusInHandler);
phoneTxt.addEventListener(FocusEvent.FOCUS_OUT, focusOutHandler);
phoneTxt.restrict = "0-9";
phoneTxt.width = 100;
phoneTxt.height = 18;
phoneTxt.background = true;
phoneTxt.border = true;
addChild(phoneTxt);

nameField.type = TextFieldType.INPUT;
nameField.x = 120;
nameField.width = 100;
nameField.height = 18;
nameField.background = true;
nameField.border = true;
addChild(nameField);

function focusInHandler(event:FocusEvent):void
{
    if (Capabilities.hasIME)
    {
        IME.enabled = false;
    }
}
function focusOutHandler(event:FocusEvent):void
{
    if (Capabilities.hasIME)
    {
        IME.enabled = true;
    }
}
```

Cet exemple crée deux champs de saisie de texte, `phoneTxt` et `nameTxt`, puis ajoute deux écouteurs d'événement au champ `phoneTxt`. Lorsque l'utilisateur place le focus sur le champ `phoneTxt`, un événement `FocusEvent.FOCUS_IN` est distribué et l'IME est désactivé. Lorsque le focus est retiré du champ `phoneTxt`, l'événement `FocusEvent.FOCUS_OUT` est distribué pour réactiver l'IME.

Ecoute des événements IME composition

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les événements IME composition sont distribués lors de la définition d'une chaîne de composition. Par exemple, si l'IME de l'utilisateur est activé et que l'utilisateur saisit une chaîne en japonais, l'événement `IMEEvent.IME_COMPOSITION` est distribué dès que l'utilisateur sélectionne la chaîne de composition. Pour écouter l'événement `IMEEvent.IME_COMPOSITION`, vous devez ajouter un écouteur d'événement à la propriété statique `ime` de la classe System (`flash.system.System.ime.addEventListener(...)`), comme le montre l'exemple suivant :

```
var inputTxt:TextField;
var outputTxt:TextField;

inputTxt = new TextField();
inputTxt.type = TextFieldType.INPUT;
inputTxt.width = 200;
inputTxt.height = 18;
inputTxt.border = true;
inputTxt.background = true;
addChild(inputTxt);

outputTxt = new TextField();
outputTxt.autoSize = TextFieldAutoSize.LEFT;
outputTxt.y = 20;
addChild(outputTxt);

if (Capabilities.hasIME)
{
    IME.enabled = true;
    try
    {
        IME.conversionMode = IMEConversionMode.JAPANESE_HIRAGANA;
    }
    catch (error:Error)
    {
        outputTxt.text = "Unable to change IME.";
    }
    System.ime.addEventListener(IMEEvent.IME_COMPOSITION, imeCompositionHandler);
}
else
{
    outputTxt.text = "Please install IME and try again.";
}

function imeCompositionHandler(event:IMEEvent):void
{
    outputTxt.text = "you typed: " + event.text;
}
```

Le code ci-dessus crée deux champs de texte et les ajoute à la liste d'affichage. Le premier champ, `inputTxt`, est un champ de saisie de texte qui permet à l'utilisateur d'entrer du texte japonais. Le second champ, `outputTxt`, est un champ de texte dynamique qui affiche des messages d'erreur à l'utilisateur ou reprend la chaîne japonaise que l'utilisateur saisit dans le champ `inputTxt`.

Claviers virtuels

Flash Player 10.2 et les versions ultérieures, AIR 2.6 et les versions ultérieures

Les périphériques mobiles, tels que téléphones et tablettes, disposent généralement d'un clavier virtuel et logiciel et non d'un clavier physique. Les classes de l'API Flash permettent d'effectuer les opérations suivantes :

- détecter le moment où le clavier virtuel apparaît et le moment où il disparaît ;
- empêcher le clavier de s'afficher ;

- déterminer la zone de la scène cachée par le clavier virtuel ;
- créer des objets interactifs qui affichent le clavier lorsqu'ils prennent le focus ; (non pris en charge par les applications AIR sous iOS)
- (AIR uniquement) désactiver le comportement de panoramique automatique afin que l'application puisse modifier son propre affichage en vue de recevoir le clavier.

Contrôle du comportement du clavier virtuel

Le moteur d'exécution ouvre automatiquement le clavier virtuel lorsque l'utilisateur appuie dans un champ de texte ou un objet interactif spécialement configuré. A l'ouverture du clavier, le moteur d'exécution respecte les conventions de la plate-forme native en matière de panoramique et de redimensionnement du contenu de l'application, afin que l'utilisateur puisse visualiser le texte au fur et à mesure qu'il le tape.

A l'ouverture du clavier, l'objet cible d'action distribue les événements suivants dans l'ordre indiqué :

Evénement `softKeyboardActivating` : distribué juste avant que le clavier ne commence à se superposer à la scène. Si vous appelez la méthode `preventDefault()` de l'objet événement distribué, le clavier virtuel ne s'ouvre pas.

Evénement `softKeyboardActivate` : distribué au terme du traitement de l'événement `softKeyboardActivating`. Lorsque l'objet cible d'action distribue cet événement, la propriété `softKeyboardRect` de l'objet Stage a été mise à jour pour refléter la partie de la scène masquée par le clavier virtuel. Il est impossible d'annuler cet événement.

Remarque : *en cas de changement de taille du clavier (si l'utilisateur modifie le type du clavier, par exemple), l'objet cible d'action distribue un second événement softKeyboardActivate.*

Evénement `softKeyboardDeactivate` : distribué lors de la fermeture du clavier virtuel, quel qu'en soit le motif. Il est impossible d'annuler cet événement.

L'exemple suivant ajoute deux objets TextField sur la scène. L'objet TextField supérieur interdit l'affichage du clavier lorsque vous touchez le champ et le ferme s'il est déjà affiché. L'objet TextField inférieur illustre le comportement par défaut. L'exemple indique les événements de clavier logiciel distribués par les deux champs de texte.

```
package
{
import flash.display.Sprite;
import flash.text.TextField;
import flash.text.TextFieldType;
import flash.events.SoftKeyboardEvent;
public class SoftKeyboardEventExample extends Sprite
{
    private var tf1:TextField = new TextField();
    private var tf2:TextField = new TextField();

    public function SoftKeyboardEventExample()
    {
        tf1.width = this.stage.stageWidth;
        tf1.type = TextFieldType.INPUT;
        tf1.border = true;
        this.addChild( tf1 );

        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, preventSoftKe
yboard );
        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, preventSoftKe
yboard );
        tf1.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, preventSoftKeyboard
);
```

```
        tf2.border = true;
        tf2.type = TextFieldType.INPUT;
        tf2.width = this.stage.stageWidth;
        tf2.y = tf1.y + tf1.height + 30;
        this.addChild( tf2 );

        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, allowSoftKeyboard );
        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, allowSoftKeyboard );
        tf2.addEventListener( SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, allowSoftKeyboard);
    }

    private function preventSoftKeyboard( event:SoftKeyboardEvent ):void
    {
            event.preventDefault();
            this.stage.focus = null; //close the keyboard, if raised
            trace( "tf1 dispatched: " + event.type + " -- " + event.triggerType );
    }
    private function allowSoftKeyboard( event:SoftKeyboardEvent )    :void
    {
            trace( "tf2 dispatched: " + event.type + " -- " + event.triggerType );
    }
}
}
```

Ajout de la prise en charge du clavier virtuel par les objets interactifs

Flash Player 10.2 et les versions ultérieures, AIR 2.6 et les versions ultérieures (non pris en charge sous iOS)

En temps normal, le clavier virtuel ne s'ouvre que si l'utilisateur appuie sur un objet TextField. Vous pouvez configurer une occurrence de la classe InteractiveObject de façon à ouvrir le clavier virtuel lorsqu'il reçoit le focus.

Pour configurer une occurrence d'InteractiveObject de façon à ouvrir le clavier logiciel, définissez la propriété `needsSoftKeyboard` sur `true`. Dès que l'objet est affecté à la propriété focus de la scène, le clavier logiciel s'ouvre automatiquement. Par ailleurs, vous pouvez afficher le clavier en appelant la méthode `requestSoftKeyboard()` de l'objet InteractiveObject.

L'exemple suivant explique comment programmer un objet InteractiveObject de façon à ce qu'il se comporte comme un champ de saisie de texte. La classe TextInput décrite dans l'exemple définit la propriété `needsSoftKeyboard` de façon à ce que le clavier s'affiche lorsque cela est nécessaire. L'objet écoute alors les événements `keyDown` et insère le caractère saisi dans le champ.

Cet exemple fait appel au moteur de texte de Flash pour ajouter et afficher le texte saisi, et gère quelques événements importants. Pour plus de simplicité, cet exemple n'implémente pas un champ de texte complet.

```
package  {
    import flash.geom.Rectangle;
    import flash.display.Sprite;
    import flash.text.engine.TextElement;
    import flash.text.engine.TextBlock;
    import flash.events.MouseEvent;
    import flash.events.FocusEvent;
    import flash.events.KeyboardEvent;
    import flash.text.engine.TextLine;
    import flash.text.engine.ElementFormat;
    import flash.events.Event;

    public class TextInput extends Sprite
    {

        public var text:String = " ";
        public  var textSize:Number = 24;
        public var textColor:uint = 0x000000;
        private var _bounds:Rectangle = new Rectangle( 0, 0, 100, textSize );
        private var textElement: TextElement;
        private var textBlock:TextBlock = new  TextBlock();

        public function TextInput( text:String = "" )
        {
            this.text = text;
            this.scrollRect = _bounds;
            this.focusRect= false;

            //Enable keyboard support
            this.needsSoftKeyboard = true;
            this.addEventListener(MouseEvent.MOUSE_DOWN, onSelect);
            this.addEventListener(FocusEvent.FOCUS_IN, onFocusIn);
            this.addEventListener(FocusEvent.FOCUS_OUT, onFocusOut);

            //Setup text engine
            textElement = new TextElement( text, new ElementFormat( null, textSize, textColor ) );
            textBlock.content = textElement;
            var firstLine:TextLine = textBlock.createTextLine( null, _bounds.width - 8 );
            firstLine.x = 4;
            firstLine.y = 4 + firstLine.totalHeight;
            this.addChild( firstLine );

        }

        private function onSelect( event:MouseEvent ):void
        {
            stage.focus = this;
        }
        private function onFocusIn( event:FocusEvent ):void
        {
            this.addEventListener( KeyboardEvent.KEY_DOWN, onKey );
        }

        private function onFocusOut( event:FocusEvent ):void
        {
            this.removeEventListener( KeyboardEvent.KEY_UP, onKey );
        }
```

```
        private function onKey( event:KeyboardEvent ):void
        {
            textElement.replaceText( textElement.text.length, textElement.text.length,
String.fromCharCode( event.charCode ) );
            updateText();
        }
        public function set bounds( newBounds:Rectangle ):void
        {
            _bounds = newBounds.clone();
            drawBackground();
            updateText();
            this.scrollRect = _bounds;

            //force update to focus rect, if needed
            if( this.stage!= null && this.focusRect && this.stage.focus == this )
                this.stage.focus = this;
        }

        private function updateText():void
        {
            //clear text lines
            while( this.numChildren > 0 ) this.removeChildAt( 0 );

            //and recreate them
            var textLine:TextLine = textBlock.createTextLine( null, _bounds.width - 8);
            while ( textLine)
            {
                textLine.x = 4;
                if( textLine.previousLine != null )
                {
                    textLine.y = textLine.previousLine.y +
                                 textLine.previousLine.totalHeight + 2;
                }
                            else
                {
                    textLine.y = 4 + textLine.totalHeight;
                }
                this.addChild(textLine);
                textLine = textBlock.createTextLine(textLine, _bounds.width - 8 );
            }
        }

        private function drawBackground():void
        {
            //draw background and border for the field
            this.graphics.clear();
            this.graphics.beginFill( 0xededed );
            this.graphics.lineStyle( 1, 0x000000 );
            this.graphics.drawRect( _bounds.x + 2, _bounds.y + 2, _bounds.width - 4,
_bounds.height - 4);
            this.graphics.endFill();
        }
    }
}
```

La classe d'application principale suivante explique comment utiliser la classe TextInput et gérer la disposition de l'application lorsque le clavier s'affiche ou lorsque l'orientation du périphérique change. La classe principale crée un objet TextInput et définit ses limites de façon à ce qu'il occupe la scène. Cette classe ajuste la taille de l'objet TextInput lorsque le clavier logiciel s'affiche ou lorsque la taille de la scène change. Cette classe écoute les événements du clavier logiciel via l'objet TextInput et redimensionne les événements à partir de la scène. Quelle que soit la cause de l'événement, l'application détermine la zone visible de la scène et redimensionne le contrôle de saisie pour le remplir. Dans une application réelle, vous auriez besoin d'un algorithme de mise en forme plus sophistiqué.

```
package  {

    import flash.display.MovieClip;
    import flash.events.SoftKeyboardEvent;
    import flash.geom.Rectangle;
    import flash.events.Event;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;

    public class CustomTextField extends MovieClip {

        private var customField:TextInput = new TextInput("Input text: ");

        public function CustomTextField() {
            this.stage.scaleMode = StageScaleMode.NO_SCALE;
            this.stage.align = StageAlign.TOP_LEFT;
            this.addChild( customField );
            customField.bounds = new Rectangle( 0, 0, this.stage.stageWidth,
this.stage.stageHeight );

            //track soft keyboard and stage resize events
            customField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE,
onDisplayAreaChange );
            customField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE,
onDisplayAreaChange );
            this.stage.addEventListener( Event.RESIZE, onDisplayAreaChange );
        }

        private function onDisplayAreaChange( event:Event ):void
        {
            //Fill the stage if possible, but avoid the area covered by a keyboard
            var desiredBounds = new Rectangle( 0, 0, this.stage.stageWidth,
this.stage.stageHeight );
            if( this.stage.stageHeight - this.stage.softKeyboardRect.height <
desiredBounds.height )
                desiredBounds.height = this.stage.stageHeight -
this.stage.softKeyboardRect.height;

            customField.bounds = desiredBounds;
        }
    }
}
```

Remarque : *la scène distribue uniquement des événements resize en réponse à un changement d'orientation lorsque la propriété* `scaleMode` *est définie sur* `noScale`*. Dans d'autres modes, les dimensions de la scène ne changent pas ; le contenu est mis à l'échelle pour compenser.*

Gestion des changements d'affichage d'une application

AIR 2.6 et les versions ultérieures

Dans AIR, vous pouvez désactiver le comportement de panoramique et de redimensionnement par défaut associé à l'affichage d'un clavier logiciel en définissant l'élément `softKeyboardBehavior` dans le descripteur d'application sur `none` :

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Lorsque vous désactivez le comportement automatique, il vous incombe d'effectuer les réglages nécessaires sur l'affichage de l'application. Un événement softKeyboardActivate est distribué lors de l'affichage du clavier. Lorsque l'événement `softKeyboardActivate` est distribué, la propriété `softKeyboardRect` de la scène contient les dimensions de la scène cachée par le clavier ouvert. Utilisez ces dimensions pour déplacer ou redimensionner votre contenu de façon à ce qu'il s'affiche correctement lorsque le clavier est ouvert et que l'utilisateur effectue une saisie. (Lorsque le clavier est fermé, les dimensions du rectangle softKeyboardRect sont toutes égales à zéro.)

Lors de la fermeture du clavier, un événement `softKeyboardDeactivate` est distribué et vous pouvez réactiver l'affichage normal de l'application.

```
package  {
    import flash.display.MovieClip;
    import flash.events.SoftKeyboardEvent;
    import flash.events.Event;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.display.InteractiveObject;
    import flash.text.TextFieldType;
    import flash.text.TextField;

    public class PanningExample extends MovieClip {

        private var textField:TextField = new TextField();

        public function PanningExample() {
            this.stage.scaleMode = StageScaleMode.NO_SCALE;
            this.stage.align = StageAlign.TOP_LEFT;

            textField.y = this.stage.stageHeight - 201;
            textField.width = this.stage.stageWidth;
            textField.height = 200;
            textField.type = TextFieldType.INPUT;
            textField.border = true;
            textField.wordWrap = true;
            textField.multiline = true;

            this.addChild( textField );

            //track soft keyboard and stage resize events
            textField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE,
onKeyboardChange );
            textField.addEventListener(SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE,
onKeyboardChange );
            this.stage.addEventListener( Event.RESIZE, onDisplayAreaChange );
        }

        private function onDisplayAreaChange( event:Event ):void
```

```
        {
            textField.y = this.stage.stageHeight - 201;
            textField.width = this.stage.stageWidth;
        }

        private function onKeyboardChange( event:SoftKeyboardEvent ):void
        {
            var field:InteractiveObject = textField;
            var offset:int = 0;

            //if the softkeyboard is open and the field is at least partially covered
            if( (this.stage.softKeyboardRect.y != 0) && (field.y + field.height >
this.stage.softKeyboardRect.y) )
                offset = field.y + field.height - this.stage.softKeyboardRect.y;

            //but don't push the top of the field above the top of the screen
            if( field.y - offset < 0 ) offset += field.y - offset;

            this.y = -offset;
        }
    }
}
```

Remarque : *sous Android, il existe des cas où les dimensions exactes du clavier ne sont pas disponibles à partir du système d'exploitation, notamment lorsque le mode plein écran est activé. Dans ces cas, la taille est estimée. Par ailleurs, en mode paysage, le clavier IME plein écran natif est utilisé pour toutes les saisies de texte. Ce clavier IME dispose d'un champ de saisie de texte et cache la totalité de la scène. Il n'existe aucun moyen d'afficher un clavier en orientation paysage qui ne remplit pas l'écran.*

Chapitre 31 : Entrées de souris

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Votre application peut créer une interactivité en capturant les entrées de la souris et en réagissant à ces dernières. Cette section suppose que vous maîtrisez le modèle d'événement d'ActionScript 3.0. Pour plus d'informations, voir « Gestion des événements » à la page 129.

Pour plus d'informations sur l'identification du type de souris pris en charge (curseur permanent, stylet, action tactile, etc.) à l'exécution, voir « Découverte des types de saisie » à la page 576.

Voir aussi

flash.ui.Mouse

flash.events.MouseEvent

« Saisie tactile, multipoint et par mouvement » à la page 600

Capture des entrées de souris

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les clics de souris créent des événements souris qui permettent de déclencher une fonctionnalité interactive. Il est possible d'ajouter un écouteur d'événement à la scène afin d'écouter les événements de souris qui se produisent à tout endroit du fichier SWF. Vous pouvez également ajouter des écouteurs d'événement à des objets sur la scène qui héritent de InteractiveObject (par exemple, Sprite ou MovieClip) ; ces écouteurs sont déclenchés lorsque vous cliquez sur l'objet.

Comme les événements de clavier, les événements de souris se propagent vers le haut. Dans l'exemple suivant, `square` étant un enfant de la scène, l'événement est distribué à la fois du sprite `square` et de l'objet scène en cas de clic sur le carré :

```
var square:Sprite = new Sprite();
square.graphics.beginFill(0xFF0000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.addEventListener(MouseEvent.CLICK, reportClick);
square.x =
square.y = 50;
addChild(square);

stage.addEventListener(MouseEvent.CLICK, reportClick);

function reportClick(event:MouseEvent):void
{
    trace(event.currentTarget.toString() + " dispatches MouseEvent. Local coords [" +
event.localX + "," + event.localY + "] Stage coords [" + event.stageX + "," + event.stageY + "]");
}
```

Dans l'exemple précédent, notez que l'événement de souris contient des informations sur l'endroit où le clic s'est produit. Les propriétés `localX` et `localY` indiquent l'emplacement du clic sur l'enfant le plus bas de la chaîne d'affichage. Par exemple, si vous cliquez dans l'angle supérieur gauche de `square`, les coordonnées locales [0,0] sont signalées car il s'agit du point d'alignement de `square`. Autrement, les propriétés `stageX` et `stageY` se réfèrent aux coordonnées globales du clic sur la scène. Le même clic signale [50,50] pour ces coordonnées car `square` y a été déplacé. Ces deux paires de coordonnées peuvent être utiles, selon la façon dont vous souhaitez répondre à l'interaction utilisateur.

Remarque : *en mode plein écran, vous pouvez configurer l'application de façon à activer le verrouillage de la souris. Le verrouillage de la souris désactive le curseur et permet de déplacer la souris sans aucune limite. Pour plus d'informations, voir « Utilisation du mode Plein écran » à la page 173.*

L'objet MouseEvent contient aussi les propriétés booléennes `altKey`, `ctrlKey` et `shiftKey`. Vous pouvez utiliser ces propriétés pour vérifier si l'utilisateur appuie également sur la touche Alt, Ctrl ou Maj au moment du clic de la souris.

Glissement de sprites sur la scène

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez autoriser les utilisateurs à faire glisser un objet Sprite sur la scène à l'aide de la méthode `startDrag()` de la classe Sprite. Le code suivant en est un exemple :

```
import flash.display.Sprite;
import flash.events.MouseEvent;

var circle:Sprite = new Sprite();
circle.graphics.beginFill(0xFFCC00);
circle.graphics.drawCircle(0, 0, 40);

var target1:Sprite = new Sprite();
target1.graphics.beginFill(0xCCFF00);
target1.graphics.drawRect(0, 0, 100, 100);
target1.name = "target1";

var target2:Sprite = new Sprite();
target2.graphics.beginFill(0xCCFF00);
target2.graphics.drawRect(0, 200, 100, 100);
target2.name = "target2";

addChild(target1);
addChild(target2);
addChild(circle);

circle.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown)

function mouseDown(event:MouseEvent):void
{
    circle.startDrag();
}
circle.addEventListener(MouseEvent.MOUSE_UP, mouseReleased);

function mouseReleased(event:MouseEvent):void
{
    circle.stopDrag();
    trace(circle.dropTarget.name);
}
```

Pour plus d'informations, voir la section consacrée à la création d'une interaction de type glissement de souris dans « Modification de la position » à la page 180.

Opération glisser-déposer dans AIR

Vous pouvez activer la prise en charge d'opérations glisser-déposer dans Adobe AIR pour permettre aux utilisateurs de faire glisser des données vers l'application et en dehors de celle-ci. Pour plus d'informations, voir « Opération glisser-déposer dans AIR » à la page 629.

Personnalisation du curseur de la souris

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le curseur de la souris (pointeur de la souris) peut être masqué ou remplacé pour tout objet d'affichage de la scène. Pour masquer ce curseur, appelez la méthode `Mouse.hide()`. Personnalisez le curseur en appelant `Mouse.hide()`, en écoutant l'événement `MouseEvent.MOUSE_MOVE` en provenance de la scène et en définissant les coordonnées d'un objet d'affichage (votre curseur personnalisé) sur les propriétés `stageX` et `stageY` de l'événement. L'exemple suivant illustre une exécution de base de cette tâche :

```
var cursor:Sprite = new Sprite();
cursor.graphics.beginFill(0x000000);
cursor.graphics.drawCircle(0,0,20);
cursor.graphics.endFill();
addChild(cursor);

stage.addEventListener(MouseEvent.MOUSE_MOVE,redrawCursor);
Mouse.hide();

function redrawCursor(event:MouseEvent):void
{
    cursor.x = event.stageX;
    cursor.y = event.stageY;
}
```

Exemple d'entrée de souris : WordSearch

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cet exemple illustre l'interaction utilisateur en gérant des événements de souris. Les utilisateurs créent autant de mots que possible à partir d'une grille aléatoire de lettres, en se déplaçant horizontalement ou verticalement dans la grille, mais en n'utilisant jamais deux fois la même lettre. Cet exemple illustre les fonctions suivantes d'ActionScript 3.0 :

- Elaboration dynamique d'une grille de composants
- Réponse aux événements souris
- Suivi du score en fonction de l'interaction utilisateur

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application WordSearch se trouvent dans le dossier Samples/WordSearch. L'application se compose des fichiers suivants :

Fichier	Description
WordSearch.as	Classe comportant les principales fonctionnalités de l'application.
WordSearch.fla ou WordSearch.mxml	Le fichier d'application principal pour Flex (MXML) ou Flash (FLA).
dictionary.txt	Un fichier utilisé pour déterminer si les mots sont écrits correctement.

Chargement d'un dictionnaire

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour créer un jeu qui implique la recherche de mots, un dictionnaire est nécessaire. L'exemple inclut un fichier texte appelé dictionary.txt, qui contient une liste de mots séparés par des retours à la ligne. Après avoir créé un tableau appelé `words`, la fonction `loadDictionary()` demande ce fichier. Une fois chargé, celui-ci devient une longue chaîne. Vous pouvez convertir cette chaîne en un tableau de mots grâce à la méthode `split()`, qui s'arrête à chaque occurrence d'un retour à la ligne (code de caractère 10) ou d'une nouvelle ligne (code de caractère 13). Cette analyse a lieu dans la fonction `dictionaryLoaded()` :

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

Création de l'interface utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une fois les mots stockés, vous pouvez élaborer l'interface utilisateur. Créez deux occurrences de bouton : l'une permet de soumettre un mot, l'autre d'effacer le mot en cours de saisie. Dans chaque cas, vous devez répondre à la saisie utilisateur en écoutant l'événement `MouseEvent.CLICK` que le bouton diffuse et en appelant ensuite une fonction. Dans la fonction `setupUI()`, ce code crée les écouteurs sur les deux boutons :

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

Génération d'un tableau de jeu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le tableau de jeu est une grille de lettres aléatoires. Dans la fonction `generateBoard()`, une grille en deux dimensions est créée en imbriquant une boucle dans une autre. La première boucle incrémente les lignes et la seconde le nombre totale de colonnes par ligne. Chaque cellule créée par ces lignes et ces colonnes contient un bouton qui représente une lettre sur le tableau.

```
private function generateBoard(startX:Number, startY:Number, totalRows:Number,
totalCols:Number, buttonSize:Number):void
{
    buttons = new Array();
    var colCounter:uint;
    var rowCounter:uint;
    for (rowCounter = 0; rowCounter < totalRows; rowCounter++)
    {
        for (colCounter = 0; colCounter < totalCols; colCounter++)
        {
            var b:Button = new Button();
            b.x = startX + (colCounter*buttonSize);
            b.y = startY + (rowCounter*buttonSize);
            b.addEventListener(MouseEvent.CLICK, letterClicked);
            b.label = getRandomLetter().toUpperCase();
            b.setSize(buttonSize,buttonSize);
            b.name = "buttonRow"+rowCounter+"Col"+colCounter;
            addChild(b);

            buttons.push(b);
        }
    }
}
```

Même si un écouteur est ajouté pour un événement `MouseEvent.CLICK` sur une seule ligne (car il est dans une boucle `for`), il est affecté à chaque occurrence de bouton. Chaque bouton reçoit un nom dérivé de la position de sa ligne et de sa colonne, ce qui permet de référencer facilement la ligne et la colonne de chaque bouton ultérieurement dans le code.

Création de mots à partir de la saisie utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les mots peuvent être écrits en sélectionnant des lettres adjacentes verticalement ou horizontalement, mais jamais en utilisant deux fois la même lettre. Chaque clic génère un événement de souris au niveau duquel le mot qu'écrit l'utilisateur doit être vérifié pour s'assurer qu'il se poursuit correctement à partir de lettres cliquées précédemment. Si ce n'est pas le cas, le mot précédent est supprimé et un nouveau est commencé. Cette vérification s'effectue dans la méthode `isLegalContinuation()`.

```
private function isLegalContinuation(prevButton:Button, currButton:Button):Boolean
{
    var currButtonRow:Number = Number(currButton.name.charAt(currButton.name. indexOf("Row") +
3));
    var currButtonCol:Number = Number(currButton.name.charAt(currButton.name.indexOf("Col") +
3));
    var prevButtonRow:Number = Number(prevButton.name.charAt(prevButton.name.indexOf("Row") +
3));
    var prevButtonCol:Number = Number(prevButton.name.charAt(prevButton.name.indexOf("Col") +
3));

    return ((prevButtonCol == currButtonCol && Math.abs(prevButtonRow - currButtonRow) <= 1) ||
             (prevButtonRow == currButtonRow && Math.abs(prevButtonCol - currButtonCol) <= 1));
}
```

Les méthodes `charAt()` et `indexOf()` de la classe String récupèrent les lignes et les colonnes du bouton sur lequel l'utilisateur clique actuellement et celui sur lequel il vient de cliquer. La méthode `isLegalContinuation()` renvoie `true` si la ligne ou la colonne est la même et que la ligne ou la colonne qui a changé se trouve à un seul incrément de la précédente. Si vous souhaitez modifier les règles du jeu et autoriser une lecture diagonale, vous pouvez supprimer la vérification de la ligne ou de la colonne inchangée. Dans ce cas, la ligne finale serait la suivante :

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol -
currButtonCol) <= 1));
```

Vérification des mots soumis

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour terminer le code pour le jeu, des mécanismes permettant de vérifier des mots soumis et de gérer le score sont nécessaires. La méthode `searchForWord()` permet ces deux opérations :

```
private function searchForWord(str:String):Number
{
    if (words && str)
    {
        var i:uint = 0
        for (i = 0; i < words.length; i++)
        {
            var thisWord:String = words[i];
            if (str == words[i])
            {
                return i;
            }
        }
        return -1;
    }
    else
    {
        trace("WARNING: cannot find words, or string supplied is null");
    }
    return -1;
}
```

Cette fonction analyse en boucle tous les mots du dictionnaire. Si le mot de l'utilisateur correspond à un mot du dictionnaire, sa position dans le dictionnaire est renvoyée. La méthode `submitWord()` vérifie la réponse et met à jour le score si la position est valide.

Personnalisation

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il existe plusieurs constantes au début de la classe. Vous pouvez modifier ce jeu en changeant ces variables. Il est par exemple possible de modifier le temps de jeu disponible en augmentant la variable `TOTAL_TIME`. Si vous augmentez légèrement la variable `PERCENT_VOWELS`, vous pouvez accroître la probabilité de trouver des mots.

Chapitre 32 : Saisie tactile, multipoint et par mouvement

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Les fonctions de gestions des événements tactiles de la plate-forme Flash incluent la saisie par point de contact unique ou multiple sur les périphériques tactiles. Le moteur d'exécution de Flash traite par ailleurs les événements qui combinent les points tactiles multiples et un geste pour créer un mouvement. En d'autres termes, les moteurs d'exécution de Flash interprètent deux types de saisie :

Saisie tactile Saisie par le biais d'un périphérique à point unique tel un doigt, un stylet ou un autre outil sur un périphérique tactile. Certains périphériques prennent en charge plusieurs points de contacts simultanés par le biais d'un doigt ou d'un stylet.

Saisie multipoint Saisie caractérisée par plusieurs points de contact simultanés.

Mouvement Saisie interprétée par un périphérique ou un système d'exploitation en réponse à un ou plusieurs événements tactiles. Un utilisateur tourne simultanément deux doigts, par exemple, et le périphérique ou le système d'exploitation interprète cette saisie tactile comme un mouvement de rotation. Certains mouvements sont exécutés à l'aide d'un doigt ou d'un point tactile, tandis que d'autres mouvements en requièrent plusieurs. Le périphérique ou le système d'exploitation détermine le type de mouvement à affecter à la saisie.

Selon le périphérique de l'utilisateur, la saisie tactile et par mouvement peut être une saisie multipoint. ActionScript propose une API de gestion des événements tactiles, des événements de mouvement et des événements tactiles suivis individuellement dans le cadre de la saisie multipoint.

Remarque : *selon le périphérique et le système d'exploitation utilisés, l'écoute des événements tactiles et de mouvement sollicite parfois un volume important de ressources de traitement (équivalent au rendu de plusieurs images par seconde). Il est souvent préférable de faire appel à des événements de souris si les fonctionnalités complémentaires assurées par la saisie tactile ou les mouvements ne sont pas nécessaires. Si vous utilisez des événements tactiles ou de mouvement, envisagez de réduire le nombre de modifications graphiques susceptibles de se produire, en particulier si ces événements peuvent être distribués rapidement, comme cela se produit lors d'une opération de type panoramique, rotation ou zoom. Vous pourriez, par exemple, arrêter l'animation au sein d'un composant pendant que l'utilisateur le redimensionne par le biais d'un mouvement de zoom.*

Voir aussi

flash.ui.Multitouch

flash.events.TouchEvent

flash.events.GestureEvent

flash.events.TransformGestureEvent

flash.events.GesturePhase

flash.events.PressAndTapGestureEvent

Paul Trani : Touch Events and Gestures on Mobile (disponible en anglais uniquement)

Mike Jones: Virtual Game Controllers (disponible en anglais uniquement)

Principes de base de la saisie tactile

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Lorsque la plate-forme Flash s'exécute dans un environnement qui prend en charge la saisie tactile, les occurrences d'InteractiveObject peuvent écouter les événements tactiles et les gestionnaires d'appel. En règle générale, les événements tactiles, multipoint et de mouvement sont traités comme tout autre événement en ActionScript (pour obtenir des informations de base sur la gestion des événements en ActionScript, voir « Gestion des événements » à la page 129).

Toutefois, pour que le moteur d'exécution de Flash puisse interpréter une action tactile ou un mouvement, il doit s'exécuter dans un environnement matériel et logiciel qui prend en charge la saisie tactile ou multipoint. Pour visualiser un graphique qui compare les différents types d'écran tactile, voir « Découverte des types de saisie » à la page 576. Par ailleurs, si le moteur d'exécution s'exécute dans une application conteneur (un navigateur, par exemple), celle-ci transmet la saisie au moteur d'exécution. Dans certains cas, l'environnement actuel caractérisé par le matériel et le système d'exploitation prend en charge la saisie multipoint, mais le navigateur qui contient le moteur d'exécution interprète la saisie et ne la transmet pas à ce dernier. Il peut également ignorer totalement la saisie.

Le diagramme suivant illustre le flux des saisies de l'utilisateur au moteur d'exécution :

Flux des saisies de l'utilisateur au moteur d'exécution de la plate-forme Flash

L'API ActionScript de développement des applications tactiles comprend heureusement des classes, méthodes et propriétés destinées à déterminer la prise en charge des saisies tactiles ou multipoint dans l'environnement d'exécution. Vous utilisez l'« API de découverte » chargée de la gestion des événements tactiles pour déterminer la prise en charge de la saisie tactile.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la programmation d'une application de gestion des événements tactile :

API de découverte Méthodes et propriétés destinées à vérifier si l'environnement d'exécution prend en charge les événements tactiles et différents modes de saisie.

Evénement tactile Action de saisie exécutée sur un périphérique tactile basée sur un point de contact unique.

Point tactile Point de contact associé à un événement tactile unique. Même si un périphérique ne prend pas en charge la saisie par mouvement, il est susceptible de gérer les points tactiles simultanés multiples.

Séquence tactile Série d'événements qui représentent la durée de vie d'une action tactile unique. Ces événements comprennent un début, zéro, une ou plusieurs actions et une fin.

Evénement multipoint Action de saisie exécutée sur un périphérique tactile basée sur plusieurs points de contact (plusieurs doigts, par exemple).

Evénement de mouvement Action de saisie exécutée sur un périphérique tactile qui trace un mouvement complexe. Exemple de mouvement : toucher de deux doigts un écran et tracer le périmètre d'un cercle abstrait en déplaçant simultanément ces deux doigts pour indiquer une rotation.

Phases Points horaires distincts du flux d'événements (le début et la fin, par exemple).

Stylet Instrument d'interaction avec un écran tactile. Un stylet est souvent plus précis que le doigt. Certains périphériques ne gèrent qu'un type déterminé de stylet. Les périphériques qui reconnaissent la saisie par stylet risquent de ne pas gérer les points de contact multiples et simultanés ou le contact d'un doigt.

Appui-appui bref Type déterminé de mouvement de saisie multipoint caractérisé comme suit : l'utilisateur place un doigt sur un périphérique tactile, puis appuie brièvement sur ce dernier avec un autre doigt ou un périphérique de pointage. Ce mouvement simule souvent un clic droit de la souris dans les applications multipoint.

Structure de l'API de gestion de la saisie tactile

De par sa conception, l'API de gestion de la saisie tactile d'ActionScript tient compte du fait que le traitement de la saisie tactile varie selon l'environnement matériel et logiciel du moteur d'exécution de Flash. Elle traite principalement trois besoins du développement d'applications tactiles : la découverte, les événements et les phases. La coordination de ces API assure à l'utilisateur une expérience prévisible et réactive, même si le périphérique cible est inconnu lors du développement de l'application.

Découverte

L'API de découverte permet de tester l'environnement matériel et logiciel à l'exécution. Les valeurs renseignées par le moteur d'exécution déterminent les saisies tactiles dont dispose le moteur d'exécution de Flash dans le contexte actuel. Vous disposez également du jeu de propriétés et méthodes de découverte pour définir l'application de sorte qu'elle réagisse aux événements de souris (au lieu des événements tactiles lorsque certains types de saisie tactile ne sont pas pris en charge par l'environnement). Pour plus d'informations, voir « Découverte de la prise en charge des actions tactiles » à la page 603.

Evénements

A l'instar de tout autre événement, ActionScript gère les événements tactiles par le biais d'écouteurs et de gestionnaires d'événement. Toutefois, la gestion des événements tactiles doit également prendre en compte les éléments suivants :

- Une action tactile peut être interprétée de diverses façons par le périphérique ou le système d'exploitation, soit en tant que séquence d'actions tactiles, soit, globalement, en tant que mouvement.
- Une action tactile unique sur un périphérique tactile (à l'aide du doigt, d'un stylet ou d'un périphérique de pointage) distribue également toujours un événement de souris. La classe MouseEvent permet d'associer l'événement de souris aux types d'événements. Vous pouvez aussi concevoir l'application de sorte à ne réagir qu'aux événements tactiles. Vous pouvez également créer une application qui réagit aux deux types d'événements.
- Une application peut réagir à plusieurs événements tactiles simultanés et traiter séparément chacun d'eux.

L'API de découverte permet généralement de traiter de manière conditionnelle les événements gérés par l'application, ainsi que leur mode de traitement. Lorsque l'application connaît l'environnement d'exécution, elle peut appeler le gestionnaire approprié ou déterminer l'objet d'événement requis quand l'utilisateur interagit avec l'application. L'application peut également indiquer qu'une saisie spécifique ne peut pas être traitée dans l'environnement actuel et proposer à l'utilisateur une autre solution ou des informations. Pour plus d'informations, voir « Gestion des événements tactiles » à la page 604 et « Gestion des événements de mouvement » à la page 609.

Phases

Pour les applications tactiles et multipoint, les objets d'événements tactiles contiennent des propriétés permettant de suivre les phases de l'interaction utilisateur. Programmez du code Write ActionScript destiné à traiter les phases de début, de mise à jour ou de fin de la saisie utilisateur de sorte à fournir un feedback à l'utilisateur. Réagissez aux phases d'événements en modifiant l'aspect des objets visuels lorsque l'utilisateur touche et déplace le point de contact à l'écran. Vous pouvez également faire appel aux phases pour suivre des propriétés déterminées d'un mouvement au fur et à mesure que celui-ci évolue.

Pour les événements de point tactile, vérifiez la durée de l'appui de l'utilisateur sur un objet interactif donné. Une application peut suivre plusieurs phases de points tactiles simultanées à titre individuel et les traiter en conséquence.

Pour un mouvement, interprétez les informations spécifiques relatives à la transformation du mouvement lorsqu'elle se produit. Suivez les coordonnées du ou des points de contact lorsqu'ils se déplacent à l'écran.

Découverte de la prise en charge des actions tactiles

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Pour définir l'étendue de la saisie tactile traitée par l'application, faites appel aux propriétés de la classe Multitouch. Testez ensuite l'environnement pour vous assurer de la prise en charge des événements gérés par le code ActionScript. Déterminez tout d'abord le type de saisie tactile pris en charge par l'application. Les options disponibles sont touch point, gesture ou none (interprétez toute saisie tactile comme un clic de souris et n'utilisez que des gestionnaires d'événement de souris). Faites ensuite appel aux propriétés et méthodes de la classe Multitouch pour vous assurer que le moteur d'exécution tourne dans un environnement qui prend en charge la saisie tactile requise par l'application. Testez l'environnement d'exécution pour vérifier s'il prend en charge les types de saisie tactile (l'interprétation des mouvements, par exemple) et réagissez en conséquence.

Remarque : *la classe Multitouch contient des propriétés statiques, qui n'appartiennent à aucune occurrence de classe. Utilisez-les avec la syntaxe de Multitouch.property, par exemple :*

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

Définition du type de saisie

Etant donné qu'un événement tactile se compose parfois d'un grand nombre d'éléments ou de phases, le moteur d'exécution de Flash doit pouvoir identifier le type de saisie tactile à interpréter. Si le doigt se contente de toucher un écran tactile, le moteur d'exécution distribue-t-il un événement tactile ? Attend-il plutôt un mouvement ? Le moteur d'exécution suit-il l'action tactile en tant qu'événement de bouton de souris enfoncé ? Une application qui prend en charge la saisie tactile doit déterminer le type d'événement tactile géré pour le moteur d'exécution de Flash. La propriété `Multitouch.inputMode` permet de déterminer le type de saisie tactile à l'intention du moteur d'exécution. Le mode de saisie correspond à l'une des trois options suivantes :

Aucun (None) Les événements tactiles ne sont pas gérés différemment. Définissez comme suit cette option : `Multitouch.inputMode=MultitouchInputMode.NONE` et traitez la saisie à l'aide de la classe MouseEvent.

Points tactiles uniques Chaque saisie tactile est interprétée individuellement et tous les points tactiles peuvent être suivis et gérés. Définissez comme suit cette option : `Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT` et traitez la saisie à l'aide de la classe TouchEvent.

Saisie par mouvement Le périphérique ou le système d'exploitation interprète la saisie en tant que mouvement complexe du doigt à l'écran. Le périphérique ou le système d'exploitation affecte globalement le mouvement à un événement de saisie par mouvement unique. Définissez comme suit cette option :

`Multitouch.inputMode=MultitouchInputMode.GESTURE` et traitez la saisie à l'aide des classes TransformGestureEvent, PressAndTapGestureEvent ou GestureEvent.

Voir « Gestion des événements tactiles » à la page 604 pour visualiser un exemple qui fait appel à la propriété `Multitouch.inputMode` en vue de définir le type de saisie avant de traiter un événement tactile.

Test de la prise en charge de la saisie tactile

D'autres propriétés de la classe Multitouch proposent des valeurs permettant d'adapter l'application à la prise en charge de la saisie tactile dans l'environnement actuel. Le moteur d'exécution de Flash renseigne les valeurs associées au nombre de points tactiles simultanés autorisés ou aux mouvements disponibles. Si le moteur d'exécution tourne dans un environnement qui ne prend pas en charge la gestion des événements tactiles requise par l'application, proposez une autre solution à l'utilisateur. Assurez par exemple la prise en charge des événements de souris ou affichez des informations relatives aux fonctions disponibles ou non dans l'environnement actuel.

Vous disposez également de l'API de prise en charge du clavier, des actions tactiles et de la souris (voir « Découverte des types de saisie » à la page 576).

Pour plus d'informations sur les tests de compatibilité, voir « Résolution des problèmes » à la page 613.

Gestion des événements tactiles

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

ActionScript gère les événements tactiles de base comme tout autre événement (les événements de souris, par exemple). Vous pouvez écouter une série d'événements tactiles définis par les constantes de type d'événement de la classe TouchEvent.

Remarque : *dans le cas d'une saisie multipoint (toucher de plusieurs doigts un périphérique, par exemple), le premier point de contact distribue un événement de souris et un événement tactile.*

Pour gérer un événement tactile de base :

1. Activez la gestion des événements tactiles dans l'application en définissant la propriété `flash.ui.Multitouch.inputMode` sur `MultitouchInputMode.TOUCH_POINT`.
2. Associez un écouteur d'événements à une occurrence de classe qui hérite des propriétés de la classe InteractiveObject (Sprite ou TextField, par exemple).
3. Indiquez le type d'événement tactile à gérer.
4. Appelez une fonction de gestionnaire d'événement en réponse à l'événement.

Le code suivant affiche par exemple un message lorsque l'utilisateur appuie brièvement sur le carré dessiné sur mySprite sur un écran tactile :

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

Propriétés des événements tactiles

Lorsqu'il se produit un événement, un objet d'événement est créé. L'objet TouchEvent contient des informations sur l'emplacement et les conditions de l'événement tactile. Vous disposez des propriétés de l'objet d'événement pour récupérer ces informations.

Le code suivant crée par exemple un objet TouchEvent *evt*, puis affiche la propriété `stageX` de l'objet d'événement (la coordonnée x du point dans l'espace de scène où s'est produite l'action tactile) dans le champ de texte :

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
myTextField.text = evt.stageX.toString;
myTextField.y = 50;
addChild(myTextField);
}
```

Pour plus d'informations sur les propriétés disponibles via l'objet d'événement, voir la classe TouchEvent.

Remarque : *certains environnements d'exécution ne prennent pas en charge toutes les propriétés TouchEvent. Ainsi, certains périphériques tactiles ne sont pas en mesure de détecter la pression appliquée par l'utilisateur sur l'écran tactile. Ils ne prennent par conséquent pas en charge la propriété* `TouchEvent.pressure`*. Essayez de tester la prise en charge de propriétés spécifiques pour assurer le fonctionnement de l'application. Pour plus d'informations, voir « Résolution des problèmes » à la page 613.*

Phases d'un événement tactile

Vous pouvez suivre les différentes phases des événements tactiles se produisant sur un objet interactif, mais aussi en dehors, exactement comme vous le feriez pour des événements de souris. Vous pouvez également suivre les événements tactiles du début à la fin d'une interaction tactile. La classe TouchEvent comporte des valeurs de gestion des événements `touchBegin`, `touchMove` et `touchEnd`.

Vous disposez par exemple des événements `touchBegin`, `touchMove` et `touchEnd` pour proposer à l'utilisateur un feedback visuel lorsqu'il touche et déplace un objet d'affichage :

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
myTextField.width = 200;
myTextField.height = 20;
addChild(myTextField);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
function onTouchBegin(event:TouchEvent) {
    myTextField.text = "touch begin" + event.touchPointID;
}
function onTouchMove(event:TouchEvent) {
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    myTextField.text = "touch end" + event.touchPointID;
}
```

Remarque : *à l'encontre des écouteurs chargés de déplacer l'événement tactile et de mettre fin à celui-ci, l'écouteur d'événements tactiles initial est associé à mySprite. Si l'objet d'affichage précède le doigt de l'utilisateur ou le périphérique de pointage, la scène continue à écouter l'événement tactile.*

ID de point tactile

La propriété `TouchEvent.touchPointID` est un composant fondamental de la programmation d'applications qui réagissent à la saisie tactile. Le moteur d'exécution de Flash affecte à chaque point tactile une valeur `touchPointID` unique. Lorsqu'une application réagit aux phases ou au mouvement d'une saisie tactile, vérifiez la valeur `touchPointID` avant de gérer l'événement. Les méthodes de déplacement par glissement de la saisie tactile de la classe Sprite utilisent la propriété `touchPointID` en tant que paramètre pour assurer le traitement de l'occurrence de saisie correcte. La propriété `touchPointID` vérifie qu'un gestionnaire d'événement réagit au point tactile approprié. Le gestionnaire d'événement réagit sinon à n'importe quelle occurrence du type d'événement tactile (tous les événements `touchMove`, par exemple) sur le périphérique, d'où un comportement imprévisible. La propriété est particulièrement importante lorsque l'utilisateur fait glisser des objets.

La propriété `touchPointID` permet de gérer une séquence entière d'actions tactiles. Une séquence d'actions tactile se compose d'un événement `touchBegin`, de zéro, un ou plusieurs événements `touchMove` et d'un événement `touchEnd`, qui possèdent tous la même valeur `touchPointID`.

L'exemple suivant définit une variable *touchMoveID*, qui vérifie que la valeur `touchPointID` est correcte avant de réagir à un événement de mouvement tactile. Si tel n'est pas le cas, d'autres actions tactiles déclenchent également le gestionnaire d'événement. Notez que les écouteurs associés aux phases de déplacement et de fin sont situés sur la scène, plutôt que l'objet d'affichage. La scène écoute les phases de déplacement et de fin au cas où l'action tactile de l'utilisateur dépasserait les limites de l'objet d'affichage.

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
addChild(myTextField);
myTextField.width = 200;
myTextField.height = 20;
var touchMoveID:int = 0;

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
function onTouchBegin(event:TouchEvent) {
    if(touchMoveID != 0) {
        myTextField.text = "already moving. ignoring new touch";
        return;
    }
    touchMoveID = event.touchPointID;

    myTextField.text = "touch begin" + event.touchPointID;
    stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
}
function onTouchMove(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch";
        return;
    }
    mySprite.x = event.stageX;
    mySprite.y = event.stageY;
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch end";
        return;
    }
    touchMoveID = 0;
    stage.removeEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.removeEventListener(TouchEvent.TOUCH_END, onTouchEnd);
    myTextField.text = "touch end" + event.touchPointID;
}
```

Toucher-glisser

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Deux méthodes ont été ajoutées à classe Sprite en vue de mieux prendre en charge les applications tactiles qui gèrent la saisie des points tactiles : `Sprite.startTouchDrag()` et `Sprite.stopTouchDrag()`. Le comportement de ces méthodes est identique à celui de `Sprite.startDrag()` et `Sprite.stopDrag()` en matière d'événements de souris. Notez toutefois que les méthodes `Sprite.startTouchDrag()` et `Sprite.stopTouchDrag()` gèrent toutes deux les valeurs `touchPointID` en tant que paramètres.

Le moteur d'exécution affecte la valeur `touchPointID` à l'objet d'événement pour un événement tactile. Cette valeur permet de réagir à un point tactile spécifique si l'environnement prend en charge les points tactiles multiples et simultanés (même s'il ne gère pas les mouvements). Pour plus d'informations sur la propriété `touchPointID`, voir « ID de point tactile » à la page 606.

Le code suivant illustre un gestionnaire d'événement start drag simple et un gestionnaire d'événement stop drag associé à un événement tactile. La variable *bg* est un objet d'affichage qui contient *mySprite* :

```
mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(e:TouchEvent) {
    e.target.startTouchDrag(e.touchPointID, false, bg.getRect(this));
    trace("touch begin");
}

function onTouchEnd(e:TouchEvent) {
    e.target.stopTouchDrag(e.touchPointID);
    trace("touch end");
}
```

Le code suivant illustre un exemple plus avancé qui combine glissement et phases d'événement tactile :

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(evt:TouchEvent) {
    evt.target.startTouchDrag(evt.touchPointID);
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}

function onTouchMove(evt:TouchEvent) {
    evt.target.alpha = 0.5;
}

function onTouchEnd(evt:TouchEvent) {
    evt.target.stopTouchDrag(evt.touchPointID);
    evt.target.width = 40;
    evt.target.height = 40;
    evt.target.alpha = 1;
}
```

Gestion des événements de mouvement

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

Traitez les événements de mouvement comme tout événement tactile de base. Vous pouvez écouter une série d'événements de mouvement définis par les constantes de type d'événement dans la classe TransformGestureEvent, la classe GestureEvent et la classe PressAndTapGestureEvent.

Pour gérer un événement tactile de mouvement :

1. Activez la prise en charge de la saisie par mouvement dans l'application en définissant la propriété `flash.ui.Multitouch.inputMode` sur `MultitouchInputMode.GESTURE`.
2. Associez un écouteur d'événements à une occurrence de classe qui hérite des propriétés de la classe InteractiveObject (Sprite ou TextField, par exemple).
3. Indiquez le type d'événement de mouvement à gérer.
4. Appelez une fonction de gestionnaire d'événement en réponse à l'événement.

Le code suivant affiche par exemple un message lorsque l'utilisateur effectue un mouvement de glissement sur le carré dessiné sur *mySprite* sur un écran tactile :

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, swipehandler);

function swipehandler(evt:TransformGestureEvent): void {
myTextField.text = "I've been swiped";
myTextField.y = 50;
addChild(myTextField);
}
```

Les événements d'appui double sont gérés de la même façon, mais font appel à la classe GestureEvent :

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, taphandler);

function taphandler(evt:GestureEvent): void {
myTextField.text = "I've been two-finger tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

Les événements d'appui-appui bref sont également gérés de la même façon, mais font appel à la classe PressAndTapGestureEvent :

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(PressAndTapGestureEvent.GESTURE_PRESS_AND_TAP, taphandler);

function taphandler(evt:PressAndTapGestureEvent): void {
myTextField.text = "I've been press-and-tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

Remarque : *certains environnements d'exécution ne prennent pas en charge tous les types d'événements GestureEvent, TransformGestureEvent et PressAndTapGestureEvent. Ainsi, certains périphériques tactiles ne sont pas en mesure de détecter un mouvement de glissement à plusieurs doigts. Les événements* `gestureSwipe` *de la classe InteractiveObject ne sont donc pas pris en charge par ces périphériques. Essayez de vérifier la prise en charge d'événements spécifiques pour assurer le fonctionnement de l'application. Pour plus d'informations, voir* « *Résolution des problèmes* » *à la page 613.*

Propriétés des événements de mouvement

Les événements de mouvement gèrent moins de propriétés que les événements tactiles de base. Vous y accédez de la même façon, par le biais de l'objet d'événement de la fonction du gestionnaire d'événement.

Le code suivant fait par exemple pivoter *mySprite* lorsque l'utilisateur effectue dessus un mouvement de rotation. Le champ de texte indique le facteur de rotation appliqué depuis le dernier mouvement (lorsque vous testez ce code, faites-le pivoter plusieurs fois pour visualiser les changements de valeur) :

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var mySpriteCon:Sprite = new Sprite();
var myTextField:TextField = new TextField();
myTextField.y = 50;
addChild(myTextField);

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(-20,-20,40,40);
mySpriteCon.addChild(mySprite);
mySprite.x = 20;
mySprite.y = 20;
addChild(mySpriteCon);

mySprite.addEventListener(TransformGestureEvent.GESTURE_ROTATE, rothandler);

function rothandler(evt:TransformGestureEvent): void {
evt.target.parent.rotationZ += evt.target.rotation;
myTextField.text = evt.target.parent.rotation.toString();
}
```

Remarque : *certains environnements d'exécution ne prennent pas en charge toutes les propriétés TransformGestureEvent. Ainsi, certains périphériques tactiles ne sont pas en mesure de détecter la rotation d'un mouvement à l'écran. Ils ne prennent par conséquent pas en charge la propriété* `TransformGestureEvent.rotation`*. Essayez de tester la prise en charge de propriétés spécifiques pour assurer le fonctionnement de l'application. Pour plus d'informations, voir* « *Résolution des problèmes* » *à la page 613.*

Phases de mouvement

Les événements de mouvement peuvent également être suivis par phases, vous permettant ainsi de suivre les propriétés au fur et à mesure que le mouvement est exécuté. Vous pouvez par exemple suivre les coordonnées x lorsqu'un objet est déplacé par le biais d'un mouvement de glissement. Utilisez ces valeurs pour tracer une ligne qui relie tous les points de son chemin, une fois le glissement terminé. Vous pouvez également modifier visuellement un objet d'affichage lorsqu'il traverse l'écran par glissement via un mouvement panoramique. Modifiez à nouveau l'objet une fois le mouvement panoramique terminé.

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {

    evt.target.localX++;

if (evt.phase==GesturePhase.BEGIN) {
    myTextField.text = "Begin";
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}
if (evt.phase==GesturePhase.UPDATE) {
    myTextField.text = "Update";
    evt.target.alpha = 0.5;
}
if (evt.phase==GesturePhase.END) {
    myTextField.text = "End";
    evt.target.width = 40;
    evt.target.height = 40;
    evt.target.alpha = 1;
}
}
```

Remarque : *la fréquence de la phase de mise à jour varie selon l'environnement du moteur d'exécution. Certaines combinaisons système d'exploitation et matériels ne retransmettent aucune mise à jour.*

La phase de mouvement correspond à « all » pour les événements de mouvement simples

Certains objets ne suivent pas chaque phase de l'événement de mouvement associé. Ils insèrent plutôt la valeur « all » dans la propriété de la phase de l'objet d'événement. Les mouvements simples tels qu'un glissement et un double appui ne suivent pas l'événement en plusieurs phases. Une fois l'événement distribué, la propriété `phase` de l'objet d'événement interactif qui écoute l'événement `gestureSwipe` ou `gestureTwoFingerTap` correspond toujours à `all` :

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, onSwipe);
mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, onTwoTap);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onSwipe(swipeEvt:TransformGestureEvent):void {
    myTextField.text = swipeEvt.phase // Output is "all"
}
function onTwoTap(tapEvt:GestureEvent):void {
    myTextField.text = tapEvt.phase // Output is "all"
}
```

Résolution des problèmes

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

La prise en charge matérielle et logicielle de la saisie tactile évolue rapidement. Le présent Guide de référence ne dresse pas la liste de tous les périphériques et combinaisons système d'exploitation/logiciels qui prennent en charge la saisie multipoint. Il contient toutefois des conseils relatifs à l'utilisation de l'API de découverte pour déterminer si le périphérique sur lequel est déployée l'application prend en charge la saisie multipoint, et propose des conseils de résolution des problèmes dans le code ActionScript.

Les moteurs d'exécution de Flash réagissent aux événements tactiles en fonction des informations transmises au moteur d'exécution par le périphérique, le système d'exploitation ou un logiciel conteneur (un navigateur, par exemple). Cette dépendance vis-à-vis de l'environnement logiciel complique la présentation de la compatibilité multipoint. L'interprétation d'un mouvement ou d'une action tactile varie parfois selon le périphérique. Une rotation consiste-t-elle en deux doigts qui pivotent simultanément ? Ou en un doigt qui trace un cercle à l'écran ? Selon l'environnement matériel et logiciel utilisé, le mouvement de rotation peut correspondre à l'un des deux cas de figure ou à une action totalement différente. Le périphérique indique la saisie utilisateur au système d'exploitation, qui transmet cette information au moteur d'exécution. Si le moteur d'exécution tourne dans un navigateur, ce dernier interprète parfois l'événement tactile ou de mouvement et ne transmet pas la saisie au moteur d'exécution. Ce comportement est similaire à celui des « touches de fonction » : vous essayez d'utiliser une combinaison de touches déterminée pour demander à Flash Player d'effectuer une action donnée dans le navigateur, au lieu de quoi ce dernier ouvre un menu.

Chaque API et classe indique si elle n'est pas compatible avec certains systèmes d'exploitation. Le cas échéant, passez en revue les entrées consacrées aux API, à commencer par la classe Multitouch : http://help.adobe.com/fr_FR/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html.

Quelques mouvements et actions tactiles courants sont décrits ci-après :

Panoramique Déplacez un doigt de gauche à droite ou de droite à gauche. Certains périphériques requièrent l'utilisation de deux doigts pour exécuter un panoramique.

Rotation Touchez l'écran de deux doigts, puis déplacez-les en formant un cercle (comme s'ils traçaient simultanément un cercle imaginaire sur une surface). Le pivot correspond au point intermédiaire entre les deux points tactiles.

Glissement Déplacez rapidement trois doigts de gauche à droite ou de droite à gauche, de haut en bas ou de bas en haut.

Zoom Touchez l'écran de deux doigts, puis séparez-les pour effectuer un zoom avant ou rapprochez-les pour effectuer un zoom arrière.

Appui-appui bref Déplacez ou appuyez un doigt, puis appuyez brièvement d'un doigt sur la surface.

La documentation de chaque périphérique passe en revue les mouvements pris en charge et leur mode d'exécution. En règle générale, selon le système d'exploitation, l'utilisateur ne doit pas toucher du doigt le périphérique entre chaque mouvement.

Si l'application ne réagit pas aux événements tactiles ou aux mouvements, vérifiez les éléments suivants :

1. Des écouteurs d'événements tactiles ou de mouvement sont-ils associés à une classe d'objet qui hérite de la classe InteractiveObject ? Seules les occurrences d'InteractiveObject peuvent écouter les événements tactiles et de mouvement.
2. Testez-vous l'application avec Flash Professional CS5 ? Si tel est le cas, essayez de publier et de tester l'application, car Flash Professional peut intercepter l'interaction.
3. Commencez par des événements simples et identifiez les événements pris en charge (l'exemple de code suivant est extrait de l'entrée consacrée à l'API de `Multitouch.inputMode`) :

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField()

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taplistener);

function taplistener(e:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

Appuyez sur le rectangle. Si cet exemple fonctionne, vous savez que l'environnement prend en charge un appui simple. Vous pouvez ensuite passer aux événements plus complexes.

Tester la prise en charge du mouvement est plus complexe. Un périphérique ou un système d'exploitation donné prend en charge toute combinaison de saisie par mouvement ou ne prend pas en charge du tout ce type de saisie.

Procédez comme suit pour tester simplement le mouvement de zoom :

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_ZOOM , onZoom);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a zoom gesture";
addChild(myTextField);

function onZoom(evt:TransformGestureEvent):void {
    myTextField.text = "Zoom is supported";
}
```

Effectuez un mouvement de zoom sur le périphérique et vérifiez si le champ de texte contient le message `Zoom is supported`. L'écouteur d'événements est ajouté à la scène pour vous permettre d'effectuer le mouvement sur toute partie de l'application test.

Procédez comme suit pour tester simplement le mouvement panoramique :

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a pan gesture";
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {
    myTextField.text = "Pan is supported";
}
```

Effectuez un mouvement panoramique sur le périphérique et vérifiez si le champ de texte contient le message `Pan is supported`. L'écouteur d'événements est ajouté à la scène pour vous permettre d'effectuer le mouvement sur toute partie de l'application test.

Certaines combinaisons de système d'exploitation et de périphérique prennent en charge les deux mouvements, d'autres un seul, d'autres aucun. En cas de doute, testez l'environnement de déploiement de l'application.

Problèmes connus

Les problèmes connus liés à la saisie tactile sont les suivants :

1. Mobile Internet Explorer sur un périphérique Windows Mobile effectue un zoom automatique sur un contenu issu d'un fichier SWF :

   Pour résoudre ce problème de comportement de zoom dans Internet Explorer, ajoutez le code suivant à la page HTML qui héberge le fichier SWF :

   ```
   <head>
   <meta name="viewport" content="width=device-width, height=device-height, initial-
   scale=1.0">
   </head>
   ```

2. Sous Windows 7 (et peut-être d'autres systèmes d'exploitation), l'utilisateur doit retirer le périphérique de pointage (ou les doigts) de l'écran entre les mouvements. Par exemple, pour exécuter une rotation et un zoom sur une image :
   - Exécutez le mouvement de rotation.
   - Retirez les doigts de l'écran.
   - Replacez les doigts sur l'écran et effectuez le mouvement de zoom.
3. Sous Windows 7 (et probablement sous d'autres systèmes d'exploitation), les mouvements de rotation et de zoom ne génèrent pas systématiquement une phase « update » ; c'est notamment le cas lorsque l'utilisateur effectue le mouvement très rapidement.
4. Windows 7 Starter Edition ne gère pas la saisie multipoint. Pour plus d'informations, voir les forums d'Adobe Labs : http://forums.adobe.com/thread/579180?tstart=0
5. Sous Mac OS 10.5.3 et les versions ultérieures, la valeur `Multitouch.supportsGestureEvents` est toujours définie sur `true`, même si le matériel ne prend pas en charge les événements de mouvement.

Chapitre 33 : Opération de copier-coller

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Utilisez les classes dans le Presse-papiers de l'interface de programmation pour copier les informations à destination et en provenance du Presse-papiers du système. Les formats des données qui peuvent être transférées en provenance ou à destination d'une application qui s'exécute dans Adobe® Flash® Player ou Adobe® AIR® sont les suivants :

- Texte
- Texte au format HTML
- Données RTF (Rich Text Format)
- Objets sérialisés
- Références d'objet (valides uniquement dans l'application d'origine)
- Bitmaps (AIR uniquement)
- Fichiers (AIR uniquement)
- Chaînes URL (AIR uniquement)

Principes de base de l'opération de copier-coller

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API copier-coller contient les classes suivantes :

Package	Classes
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

La propriété statique `Clipboard.generalClipboard` représente le Presse-papiers du système d'exploitation. La classe Clipboard fournit des méthodes de lecture et d'écriture des données dans les objets Clipboard.

Les classes HTMLLoader (dans AIR) et TextField implémentent un comportement par défaut pour les raccourcis clavier de copie et collage. Pour implémenter un comportement de raccourci copie et collage pour des composants personnalisés, vous pouvez vous mettre directement à l'écoute de ces frappes. Vous pouvez également utiliser des commandes de menu natives accompagnées d'équivalents de touches pour répondre indirectement aux frappes.

Un objet Clipboard unique peut servir à rendre disponibles des représentations diverses des mêmes informations afin d'augmenter les possibilités des autres applications à comprendre et à utiliser ces données. Par exemple, une image pourrait être incluse en tant que données d'image, un objet Bitmap sérialisé et un fichier. Le rendu des données dans un format peut être reporté de telle sorte que celui-ci ne soit pas effectivement créé tant que les données de ce format ne sont pas lues.

Lecture en provenance et écriture à destination du Presse-papiers du système

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour lire le contenu du Presse-papiers du système, appelez la méthode `getData()` de l'objet `Clipboard.generalClipboard` en lui communiquant le nom du format à lire :

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

Remarque : *un contenu qui s'exécute dans Flash Player ou dans un sandbox non applicatif d'AIR ne peut appeler que la méthode `getData()` dans un gestionnaire d'événement pour un événement `paste`. En d'autres termes, seul le code en cours d'exécution dans un sandbox d'application AIR peut appeler la méthode `getData()` hors d'un gestionnaire d'événement `paste`.*

Pour écrire dans le Presse-papiers, ajoutez les données à l'objet `Clipboard.generalClipbooard` dans un ou plusieurs formats. Toute donnée existante du même format est automatiquement écrasée. Toutefois, prenez l'habitude de vider le Presse-papiers du système avant de lui envoyer de nouvelles données. Vous vous assurez ainsi que les données superflues dans d'autres formats sont également supprimées.

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

var textToCopy:String = "Copy to clipboard.";
Clipboard.generalClipboard.clear();
Clipboard.generalClipboard.setData(ClipboardFormats.TEXT_FORMAT, textToCopy, false);
```

Remarque : *un contenu qui s'exécute dans Flash Player ou dans un sandbox non applicatif d'AIR ne peut appeler que la méthode `setData()` dans un gestionnaire d'événement utilisateur, tel un événement de clavier ou de souris, ou bien encore un événement `copy` ou `cut`. En d'autres termes, seul le code en cours d'exécution dans un sandbox d'application AIR peut appeler la méthode `setData()` hors d'un gestionnaire d'événement utilisateur.*

Opération copier-coller HTML dans AIR

Adobe AIR 1.0 et les versions ultérieures

Dans Adobe AIR, l'environnement HTML fournit son propre comportement et son lot d'événements par défaut pour l'opération de copier-coller. Seul le code en cours d'exécution dans le sandbox de l'application est en mesure d'accéder directement au Presse-papiers du système via l'objet `Clipboard.generalClipboard`. Le code JavaScript d'un sandbox hors application peut accéder au Presse-papiers via l'objet événement envoyé en réponse à l'un des événements copier-coller envoyé par un élément provenant d'un document HTML.

Les événements copier-coller sont les suivants : `copy`, `cut` et `paste`. L'objet envoyé pour ces événements fournit un accès au Presse-papiers par la propriété `clipboardData`.

Comportement par défaut

Adobe AIR 1.0 et les versions ultérieures

Par défaut, AIR copie les éléments sélectionnées en réponse à la commande copier qui peut être engendrée soit par un raccourci-clavier, soit par un menu contextuel. Au sein des zones modifiables, AIR coupe du texte en réponse à des commandes couper ou bien colle du texte au curseur ou à la sélection en réponse à la commande coller.

Pour éviter le comportement par défaut, votre gestionnaire d'événement peut appeler la méthode `preventDefault()` de l'objet événement envoyé.

Utilisation de la propriété clipboardData de l'objet événement

Adobe AIR 1.0 et les versions ultérieures

La propriété `clipboardData` de l'objet événement envoyé à la suite de l'un des événements copier-coller vous permet de lire des données du Presse-papiers et d'écrire dedans.

Pour écrire dans le Presse-papiers lors de l'exécution d'un événement copier-coller, utilisez la méthode `setData()` de l'objet `clipboardData` en faisant passer les données à copier et le type de MIME :

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

Pour accéder aux données en cours de collage, vous pouvez utiliser la méthode `getData()` de l'objet `clipboardData` en faisant passer le type de MIME du format de données. Les formats disponibles sont indiqués par la propriété `types`.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

Il est possible d'accéder à la méthode `getData()` et à la propriété `types` uniquement dans l'objet d'événement envoyé par l'événement `paste`.

L'exemple ci-dessous illustre la façon de remplacer le comportement copier-coller par défaut dans une page HTML. Le gestionnaire d'événement `copy` met en italiques le texte copié et le copie dans le Presse-papiers sous la forme de texte HTML. Le gestionnaire d'événement `cut` copie les données sélectionnées dans le Presse-papiers et les supprime du document. Le gestionnaire `paste` insère le contenu du Presse-papiers sous forme de HTML en caractères gras :

```
 <html>
<head>
    <title>Copy and Paste</title>
    <script language="javascript" type="text/javascript">
        function onCopy(event){
            var selection = window.getSelection();
            event.clipboardData.setData("text/html","<i>" + selection + "</i>");
            event.preventDefault();
        }

        function onCut(event){
             var selection = window.getSelection();
             event.clipboardData.setData("text/html","<i>" + selection + "</i>");
             var range = selection.getRangeAt(0);
             range.extractContents();

            event.preventDefault();
        }

        function onPaste(event){
            var insertion = document.createElement("b");
            insertion.innerHTML = event.clipboardData.getData("text/html");
             var selection = window.getSelection();
             var range = selection.getRangeAt(0);
             range.insertNode(insertion);
            event.preventDefault();
        }
    </script>
</head>
<body onCopy="onCopy(event)"
     onPaste="onPaste(event)"
     onCut="onCut(event)">
<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore
veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam
voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur
magni dolores eos qui ratione voluptatem sequi nesciunt.</p>
</body>
</html>
```

Formats de données Clipboard

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les formats de Clipboard décrivent les données placées dans un objet Clipboard. Flash Player ou AIR traduit automatiquement les formats de données standard entre types de données ActionScript et formats de Presse-papiers système. De surcroît, les objets application peuvent être transférés au sein des applications basées sur ActionScript et entre celles-ci à l'aide des formats définis par les applications.

Un objet Clipboard peut contenir des représentations des mêmes informations dans différents formats. Par exemple, un objet Clipboard représentant un sprite pourrait inclure un format de référence pour un usage au sein de la même application, un format sérialisé pour un usage par une autre application qui s'exécuterait dans Flash Player ou AIR, un format bitmap pour un usage par un éditeur d'images et un format liste de fichiers (peut-être avec un rendu différé pour coder un fichier PNG) pour permettre une copie ou un glissement d'une représentation du sprite vers le système de fichiers.

Formats de données standard

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les constantes qui définissent les noms des formats standard sont fournies par la classe ClipboardFormats :

Constante	Description
TEXT_FORMAT	Les données au format texte sont transposées en provenance et à destination de la classe String dans ActionScript.
HTML_FORMAT	Texte avec balisage HTML.
RICH_TEXT_FORMAT	Les données au format RTF sont transposées en provenance et à destination de la classe ByteArray d'ActionScript. Le balisage RTF n'est ni interprété ni transposé de quelque façon que ce soit.
BITMAP_FORMAT	(AIR uniquement) Les données au format bitmap sont transposées en provenance et à destination de la classe BitmapData d'ActionScript.
FILE_LIST_FORMAT	(AIR uniquement) Les données au format liste de fichiers sont transposées en provenance et à destination d'un tableau des objets File d'ActionScript.
URL_FORMAT	(AIR uniquement) Les données au format URL sont transposées en provenance et à destination de la classe String d'ActionScript.

Lorsqu'il s'agit de copier et de coller des données en réponse à un événement `copy`, `cut` ou `paste` dans un contenu HTML hébergé dans une application AIR, il faut utiliser des types MIME à la place des chaînes ClipboardFormat. Les types MIME de données valides sont les suivants :

Type MIME	Description
Texte	"text/plain"
URL	"text/uri-list"
Image bitmap	"image/x-vnd.adobe.air.bitmap"
Liste de fichiers	"application/x-vnd.adobe.air.file-list"

Remarque : *les données RTF ne sont pas disponibles depuis la propriété `clipboardData` de l'objet événement envoyé à la suite d'un événement `paste` au sein d'un contenu HTML.*

Formats de données personnalisés

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser des formats personnalisés définis dans les applications pour transférer des objets en tant que références ou des copies sérialisées. Les références ne sont valides que dans l'application d'origine. Les objets sérialisés peuvent être transférés entre applications, mais ne peuvent être utilisés qu'avec des objets qui demeurent valides lorsqu'ils sont sérialisés et désérialisés. Les objets peuvent généralement être sérialisés si leurs propriétés sont soit des types simples, soit des objets sérialisables.

Pour ajouter un objet sérialisé à un objet Clipboard, définissez le paramètre *serializable* sur `true` lorsque vous appelez la méthode `Clipboard.setData()`. Le nom du format peut être celui d'un format standard ou une chaîne arbitraire définie par votre application.

Modes de transfert

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un objet est écrit dans le Presse-papiers dans un format de données personnalisé, les données de l'objet peuvent être lues depuis le Presse-papiers soit comme une référence, soit comme une copie sérialisée de l'objet d'origine. Quatre modes de transfert déterminent si les objets sont transférés en tant que références ou en tant que copies sérialisées :

Mode de transfert	Description
ClipboardTransferModes.ORIGINAL_ONLY	Seule une référence est renvoyée. Si aucune référence n'est disponible, une valeur null est renvoyée.
ClipboardTransferModes.ORIGINAL_PREFFERED	Une référence est renvoyée, si elle est disponible. Sinon, une copie sérialisable est renvoyée.
ClipboardTransferModes.CLONE_ONLY	Seule une copie sérialisable est renvoyée. S'il n'y a pas de copie sérialisée, une valeur « null » est renvoyée.
ClipboardTransferModes.CLONE_PREFFERED	Une copie sérialisable est renvoyée, si elle est disponible. Sinon, une référence est renvoyée.

Lecture et écriture de formats de données personnalisés

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lors de l'écriture d'un objet dans le Presse-papiers, vous pouvez utiliser toute chaîne qui ne commence pas par les préfixes réservés `air:` ou `flash:` pour le paramètre *format*. Utilisez la même chaîne que le format pour lire l'objet. Les exemples suivants illustrent la façon de lire et d'écrire des objets dans le Presse-papiers.

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

Pour extraire un objet sérialisé de l'objet Clipboard (à la suite d'une opération déposer ou coller), utilisez le même nom de format et les modes de transfert `CLONE_ONLY` ou `CLONE_PREFFERED`.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

Une référence est toujours ajoutée à l'objet Clipboard. Pour extraire la référence de l'objet Clipboard (à la suite d'une opération déposer ou coller), utilisez les modes de transfert `ORIGINAL_ONLY` ou `ORIGINAL_PREFFERED` en lieu et place de la copie sérialisée.

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

Les références ne sont valides que si l'objet Clipboard provient de l'application actuelle. Utilisez le mode de transfert `ORIGINAL_PREFFERED` pour accéder à la référence lorsqu'elle devient disponible et le clone sérialisé lorsque la référence ne l'est pas.

Rendu différé

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si la création d'un format de données est coûteuse en ressources informatiques, vous pouvez procéder à un rendu différé en fournissant une fonction qui fournit les données à la demande. La fonction n'est appelée que si le récepteur de l'opération déposer ou coller demande les données dans le format différé.

La fonction de rendu est ajoutée à l'objet Clipboard à l'aide de la méthode`setDataHandler()`. La fonction doit renvoyer les données dans un format approprié. Par exemple, si vous appelez `setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)`, alors la fonction `writeText()` doit renvoyer une chaîne.

Si un format de données de même type est ajouté à l'objet Clipboard avec la méthode `setData()`, ces données priment sur la version différée (la fonction de rendu n'est jamais appelée). Selon le cas, la fonction de rendu peut être appelée à nouveau ou non si les mêmes données du Presse-papiers sont à nouveau utilisées.

Remarque : *sous Mac OS X, le rendu différé ne fonctionne qu'avec des formats de données personnalisés. Avec des formats de données standard, la fonction de rendu est appelée immédiatement.*

Collage de texte à l'aide d'une fonction de rendu différé

Flash Player 10 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple ci-dessous illustre la façon d'implémenter une fonction de rendu.

Lorsque l'utilisateur appuie sur le bouton Copier, l'application vide le Presse-papiers du système pour s'assurer qu'aucune donnée d'opérations de Presse-papiers précédentes ne subsiste. La méthode `setDataHandler()` définit alors la fonction `renderData()` comme rendu du Presse-papiers.

Lorsque l'utilisateur choisit la commande Coller du menu contextuel du champ de texte de destination, l'application accède au Presse-papiers et définit le texte de destination. Comme le format des données texte du Presse-papiers a été défini avec une fonction plutôt qu'une chaîne, le Presse-papiers appelle la fonction `renderData()`. Cette fonction `renderData()` renvoie le texte dans le texte source qui, lui, est affecté au texte de destination.

Notez que si vous éditez le texte source avant d'appuyer sur le bouton Coller, les modifications se retrouveront dans le texte collé, même lorsque l'édition survient après avoir appuyé sur le bouton Copier. Cette situation existe du fait que la fonction de rendu ne copie pas le texte source tant que l'on n'a pas appuyé sur le bouton Coller. Lorsque vous utilisez le rendu différé dans une véritable application, vous pourriez vouloir stocker ou protéger les données source de manière à éviter ce problème.

Exemple Flash

```
package {
    import flash.desktop.Clipboard;
    import flash.desktop.ClipboardFormats;
    import flash.desktop.ClipboardTransferMode;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.text.TextFieldType;
    import flash.events.MouseEvent;
    import flash.events.Event;
    public class DeferredRenderingExample extends Sprite
    {
        private var sourceTextField:TextField;
        private var destination:TextField;
        private var copyText:TextField;
        public function DeferredRenderingExample():void
        {
            sourceTextField = createTextField(10, 10, 380, 90);
            sourceTextField.text = "Neque porro quisquam est qui dolorem "
                + "ipsum quia dolor sit amet, consectetur, adipisci velit.";

            copyText = createTextField(10, 110, 35, 20);
            copyText.htmlText = "<a href='#'>Copy</a>";
            copyText.addEventListener(MouseEvent.CLICK, onCopy);

            destination = createTextField(10, 145, 380, 90);
            destination.addEventListener(Event.PASTE, onPaste);
        }
        private function createTextField(x:Number, y:Number, width:Number,
                        height:Number):TextField
        {
            var newTxt:TextField = new TextField();
            newTxt.x = x;
            newTxt.y = y;
            newTxt.height = height;
            newTxt.width = width;
            newTxt.border = true;
            newTxt.multiline = true;
            newTxt.wordWrap = true;
            newTxt.type = TextFieldType.INPUT;
            addChild(newTxt);
            return newTxt;
        }
        public function onCopy(event:MouseEvent):void
        {
            Clipboard.generalClipboard.clear();
            Clipboard.generalClipboard.setDataHandler(ClipboardFormats.TEXT_FORMAT,
                            renderData);
        }
        public function onPaste(event:Event):void
        {
```

```
            sourceTextField.text =
            Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT).toString;
        }
        public function renderData():String
        {
            trace("Rendering data");
            var sourceStr:String = sourceTextField.text;
            if (sourceTextField.selectionEndIndex >
                     sourceTextField.selectionBeginIndex)
            {
                return sourceStr.substring(sourceTextField.selectionBeginIndex,
                                    sourceTextField.selectionEndIndex);
            }
            else
            {
                return sourceStr;
            }
        }
    }
}
```

Exemple Flex

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" width="326"
height="330" applicationComplete="init()">
    <mx:Script>
    <![CDATA[
    import flash.desktop.Clipboard;
    import flash.desktop.ClipboardFormats;

    public function init():void
    {
        destination.addEventListener("paste", doPaste);
    }

    public function doCopy():void
    {
        Clipboard.generalClipboard.clear();
        Clipboard.generalClipboard.setDataHandler(ClipboardFormats.TEXT_FORMAT, renderData);
    }
    public function doPaste(event:Event):void
    {
        destination.text =
Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT).toString;
    }

    public function renderData():String{
        trace("Rendering data");
        return source.text;
    }
    ]]>
    </mx:Script>
    <mx:Label x="10" y="10" text="Source"/>
    <mx:TextArea id="source" x="10" y="36" width="300" height="100">
        <mx:text>Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur,
adipisci velit.</mx:text>
    </mx:TextArea>
    <mx:Label x="10" y="181" text="Destination"/>
    <mx:TextArea id="destination"  x="12" y="207" width="300" height="100"/>
    <mx:Button click="doCopy();" x="91" y="156" label="Copy"/>
</mx:Application>
```

Chapitre 34 : Saisie via un accéléromètre

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2 et les versions ultérieures

La classe Accelerometer distribue des événements en fonction de l'activité détectée par le capteur de mouvement du périphérique. Ces données représentent l'emplacement du périphérique ou un mouvement de ce dernier le long d'un axe tridimensionnel. Lorsque le périphérique se déplace, le capteur détecte le mouvement et renvoie ses coordonnées d'accélération. Les méthodes de la classe Accelerometer permettent de savoir si l'accéléromètre est pris en charge et de définir la fréquence de distribution des événements d'accélération.

Les axes de l'accéléromètre sont normalisés par rapport à l'orientation de l'affichage, plutôt qu'à l'orientation physique du périphérique. Toute modification de l'orientation de l'affichage par le périphérique entraîne la réorientation en conséquence des axes de l'accéléromètre. Par conséquent, l'axe y est toujours plus ou moins vertical lorsque l'utilisateur tient le téléphone dans une position verticale standard, quelle que soit la direction de rotation du téléphone. Si l'orientation automatique est désactivée (lorsque le contenu SWF d'un navigateur est affiché en mode Plain écran, par exemple), les axes de l'accéléromètre ne sont pas réorientés lors de la rotation du périphérique.

Voir aussi

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

Vérification de la prise en charge de l'accéléromètre

La propriété `Accelerometer.isSupported` permet de vérifier si l'environnement d'exécution prend en charge cette fonction :

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

Les versions du moteur d'exécution indiquées pour chaque entrée d'API peuvent accéder à la classe Accelerometer et à ses membres. L'environnement d'exécution actuel détermine toutefois la disponibilité de la fonction. Vous pouvez, par exemple, compiler du code par le biais des propriétés de la classe Accelerometer pour Flash Player 10.1, mais vous devez faire appel à la propriété `Accelerometer.isSupported` pour vérifier la disponibilité de la fonction Accelerometer sur le périphérique de l'utilisateur. Si `Accelerometer.isSupported` est défini sur `true` à l'exécution, la prise en charge de la fonction Accelerometer est active.

Détection des changements de l'accéléromètre

Pour utiliser le capteur de l'accéléromètre, instanciez un objet Accelerometer et enregistrez-vous pour recevoir les événements `update` qu'il distribue. Un événement `update` est un objet d'événement Accelerometer. L'événement possède quatre propriétés, toutes numériques :

- `accelerationX` : accélération le long de l'axe x, mesurée en g. L'axe x traverse le périphérique de gauche à droite lorsque l'utilisateur le tient droit. (La partie supérieure du périphérique est alors orientée vers le haut.) L'accélération est positive si le périphérique se déplace vers la droite.
- `accelerationY` : accélération le long de l'axe y, mesurée en g. L'axe y traverse le périphérique de bas en haut lorsque l'utilisateur le tient droit. (La partie supérieure du périphérique est alors orientée vers le haut.) L'accélération est positive si le périphérique se déplace vers le haut par rapport à cet axe.
- `accelerationZ` : accélération le long de l'axe z, mesurée en g. L'axe z est perpendiculaire à la face du périphérique. L'accélération est positive si vous déplacez le périphérique de sorte que sa face soit orientée vers le haut. L'accélération est négative si la face du périphérique est orientée vers le sol.
- `timestamp` : nombre de millisecondes au moment où se produit l'événement après l'initialisation du moteur d'exécution.

1 g correspond à l'accélération standard due à la gravité, soit environ 9,8 m/s^2.

Exemple de base qui affiche les données de l'accéléromètre dans un champ de texte :

```
var accl:Accelerometer;
if (Accelerometer.isSupported)
{
    accl = new Accelerometer();
    accl.addEventListener(AccelerometerEvent.UPDATE, updateHandler);
}
else
{
    accTextField.text = "Accelerometer feature not supported";
}
function updateHandler(evt:AccelerometerEvent):void
{
    accTextField.text = "acceleration X: " + evt.accelerationX.toString() + "\n"
            + "acceleration Y: " + evt.accelerationY.toString() + "\n"
            + "acceleration Z: " + evt.accelerationZ.toString()
}
```

Pour exploiter cet exemple, veillez à créer le champ de texte `accTextField` et à l'ajouter à la liste d'affichage avant d'utiliser le code.

Vous pouvez ajuster l'intervalle de temps qui sépare les événements de l'accéléromètre en appelant la méthode `setRequestedUpdateInterval()` de l'objet Accelerometer. Cette méthode ne gère qu'un seul paramètre, `interval`, qui correspond à la fréquence de mise à jour requise, en millisecondes :

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

L'intervalle réel entre les mises à jour de l'accéléromètre peut être supérieur ou inférieur à cette valeur. Toute modification de l'intervalle de mise à jour affecte l'ensemble des écouteurs enregistrés. Si vous n'appelez pas la méthode `setRequestedUpdateInterval()`, l'application reçoit des mises à jour à la fréquence définie par défaut sur le périphérique.

Les données de l'accéléromètre ne sont pas d'une précision absolue. Vous pouvez calculer la moyenne mobile des données récentes pour lisser les données. Ainsi, l'exemple suivant prend en compte les données récentes issues de l'accéléromètre et les données en cours pour arrondir le résultat :

```
var accl:Accelerometer;
var rollingX:Number = 0;
var rollingY:Number = 0;
var rollingZ:Number = 0;
const FACTOR:Number = 0.25;

if (Accelerometer.isSupported)
{
    accl = new Accelerometer();
    accl.setRequestedUpdateInterval(200);
    accl.addEventListener(AccelerometerEvent.UPDATE, updateHandler);
}
else
{
    accTextField.text = "Accelerometer feature not supported";
}
function updateHandler(event:AccelerometerEvent):void
{
    accelRollingAvg(event);
    accTextField.text = rollingX + "\n" +  rollingY + "\n" + rollingZ + "\n";
}

function accelRollingAvg(event:AccelerometerEvent):void
{
    rollingX = (event.accelerationX * FACTOR) + (rollingX * (1 - FACTOR));
    rollingY = (event.accelerationY * FACTOR) + (rollingY * (1 - FACTOR));
    rollingZ = (event.accelerationZ * FACTOR) + (rollingZ * (1 - FACTOR));
}
```

Le calcul de la moyenne mobile n'est toutefois désirable que si la fréquence de mise à jour de l'accéléromètre est élevée.

Chapitre 35 : Opération glisser-déposer dans AIR

Adobe AIR 1.0 et les versions ultérieures

Les classes de l'API glisser-déposer d'Adobe® AIR™ permettent de prendre en charge les mouvements glisser-déposer dans l'interface utilisateur. Dans ce contexte, un *mouvement* est une action effectuée par l'utilisateur, gérée à la fois par le système d'exploitation et votre application, qui exprime son intention de copier, déplacer ou lier des informations. Il se produit un mouvement de *glissement vers l'extérieur* lorsque l'utilisateur fait glisser un objet hors d'un composant ou d'une application. Il se produit un mouvement de *glissement vers l'intérieur* lorsque l'utilisateur fait glisser un objet externe vers un composant ou une application.

L'API glisser-déposer permet à un utilisateur de faire glisser des données d'une application à l'autre ou d'un composant d'application à l'autre. Parmi les formats de transfert pris en charge figurent :

- Images bitmap
- Fichiers
- Texte au format HTML
- Texte
- Données RTF (Rich Text Format)
- URL
- Fichiers promis
- Objets sérialisés
- Références aux objets (valables uniquement dans leur application d'origine)

Principes de base des opérations glisser-déposer dans AIR

Adobe AIR 1.0 et les versions ultérieures

Pour obtenir une explication rapide de l'utilisation des opérations glisser-déposer dans une application AIR et des exemples de code correspondants, voir les articles de démarrage rapide dans Adobe Developer Connection :

- Prise en charge des opérations glisser-déposer et copier-coller (Flex)
- Prise en charge des opérations glisser-déposer et copier-coller (Flash)

L'API glisser-déposer contient les classes suivantes.

Package	Classes
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise Les constantes utilisées dans l'API glisser-déposer sont définies dans les classes suivantes : • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

Décomposition des mouvements glisser-déposer

Le mouvement glisser-déposer est composé de trois étapes, comme suit :

Initialisation *Un utilisateur initialise une opération de glisser-déposer en faisant glisser un composant ou un élément de composant tout en maintenant enfoncé le bouton de la souris.* Le composant source de l'élément glissé porte généralement le nom d'initiateur du glissement et distribue les événements `nativeDragStart` et `nativeDragComplete`. Une application Adobe AIR démarre une opération de glissement en appelant la méthode `NativeDragManager.doDrag()` en réponse à un événement `mouseDown` ou `mouseMove`.

Si l'opération de glissement commence à l'extérieur d'une application AIR, aucun objet initiateur ne déclenche les événements `nativeDragStart` ou `nativeDragComplete`.

Glissement *Tout en maintenant enfoncé le bouton de la souris, l'utilisateur déplace le curseur vers un autre composant, une autre application ou le bureau.* Pendant la durée du glissement, l'objet initiateur distribue des événements `nativeDragUpdate`. (Cet événement n'est toutefois pas distribué dans AIR pour Linux.) Lorsque l'utilisateur place la souris sur une cible de dépôt potentielle dans une application AIR, la cible distribue un événement `nativeDragEnter`. Le gestionnaire d'événement peut inspecter l'objet événement pour déterminer si les données glissées sont disponibles dans un format géré par la cible. Si tel est le cas, il autorise l'utilisateur à déposer les données sur la cible en appelant la méthode `NativeDragManager.acceptDragDrop()`.

Tant que le mouvement de glissement survole un objet interactif, celui-ci distribue des événements `nativeDragOver`. Lorsque le mouvement de glissement quitte l'objet interactif, celui-ci distribue un événement `nativeDragExit`.

Dépôt *L'utilisateur relâche la souris sur une cible de dépôt appropriée.* Si la cible est un composant ou une application AIR, l'objet cible déclenche un événement `nativeDragDrop`. Le gestionnaire d'événement peut accéder aux données transférées à partir de l'objet événement. Si la cible n'est ni un composant, ni une application AIR, le système d'exploitation ou une autre application gère le dépôt. Dans les deux cas de figure, l'objet initiateur distribue un événement `nativeDragComplete` (sous réserve que le glissement ait débuté à partir d'AIR).

La classe NativeDragManager contrôle les mouvements de glissement vers l'intérieur et l'extérieur. Tous les membres de la classe NativeDragManager étant statiques, ils ne créent pas d'occurrence de cette dernière.

Objet Clipboard

Les données glissées vers une application ou un composant ou à partir de ces derniers sont contenues dans un objet Clipboard. Un objet Clipboard unique peut proposer plusieurs représentations des mêmes informations pour augmenter les chances qu'une autre application comprenne et utilise les données. Une image peut par exemple être incluse en tant que données d'image, objet Bitmap sérialisé ou fichier. Le rendu des données dans un format peut être confié à une fonction de rendu qui n'est pas appelée tant que les données n'ont pas été lues.

Une fois le mouvement de glissement démarré, il n'est possible d'accéder à l'objet Clipboard qu'à partir d'un gestionnaire associé aux événements `nativeDragEnter`, `nativeDragOver` et `nativeDragDrop`. Lorsque le mouvement de glissement est terminé, l'objet Clipboard ne peut être ni lu ni réutilisé.

Un objet application peut être transféré sous forme de référence et d'objet sérialisé. Les références sont valables uniquement dans leur application d'origine. Les transferts d'objets sérialisés sont valides d'une application AIR à une autre, mais sont réservés aux objets qui demeurent valides lorsqu'ils sont sérialisés et désérialisés. Les objets sérialisés sont convertis au format AMF3 (Action Message Format for ActionScript 3), qui permet de transférer des données sous forme de chaînes.

Utilisation de la structure Flex

Dans la plupart des cas, il est recommandé d'utiliser l'API glisser-déposer d'Adobe® Flex™ lors de la création d'applications Flex. La structure Flex propose un jeu de fonctions équivalentes lorsqu'une application Flex est exécutée dans AIR (elle utilise la classe NativeDragManager d'AIR en interne). Flex gère également un jeu plus limité de fonctions lorsqu'une application ou un composant s'exécute dans l'environnement plus restrictif d'un navigateur. Il est impossible d'utiliser les classes AIR dans des composants ou applications qui tournent en dehors de l'environnement d'exécution AIR.

Prise en charge du mouvement de glissement vers l'extérieur

Adobe AIR 1.0 et les versions ultérieures

Pour prendre en charge le mouvement de glissement vers l'extérieur, vous devez créer un objet Clipboard en réponse à un événement `mouseDown` et l'envoyer à la méthode `NativeDragManager.doDrag()`. Votre application peut alors écouter l'événement `nativeDragComplete` sur l'objet initiateur pour déterminer la marche à suivre lorsque l'utilisateur termine ou abandonne le mouvement.

Préparation des données à transférer

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour préparer les données ou un objet à faire glisser, créez un objet Clipboard et ajoutez les informations à transférer dans un ou plusieurs formats. Vous disposez des formats de données standard pour transmettre des données susceptibles d'être converties automatiquement en formats Presse-papiers natifs, ainsi que des formats définis par l'application pour transmettre des objets.

Si la conversion d'informations à transférer dans un format déterminé mobilise un volume élevé de ressources de calcul, vous pouvez indiquer le nom d'une fonction de gestionnaire qui exécutera la conversion. La fonction est appelée sous réserve que le composant ou l'application qui reçoit les données lise le format correspondant.

Pour plus d'informations sur les formats du Presse-papiers, voir « Formats de données Clipboard » à la page 619.

L'exemple suivant illustre la création d'un objet Clipboard qui comporte une image bitmap en plusieurs formats : un objet Bitmap, un format d'image bitmap natif et un format de liste de fichiers contenant le fichier source de l'image bitmap :

```
import flash.desktop.Clipboard;
import flash.display.Bitmap;
import flash.filesystem.File;
public function createClipboard(image:Bitmap, sourceFile:File):Clipboard{
    var transfer:Clipboard = new Clipboard();
    transfer.setData("CUSTOM_BITMAP", image, true); //Flash object by value and by reference
    transfer.setData(ClipboardFormats.BITMAP_FORMAT, image.bitmapData, false);
    transfer.setData(ClipboardFormats.FILE_LIST_FORMAT, new Array(sourceFile), false);
    return transfer;
}
```

Démarrage d'une opération de glissement vers l'extérieur

Adobe AIR 1.0 et les versions ultérieures

Pour démarrer une opération de glissement, appelez la méthode `NativeDragManager.doDrag()` en réponse à un événement mouse down. La méthode `doDrag()` est statique et gère les paramètres suivants :

Paramètre	Description
initiator	Objet où débute le glissement, qui distribue les événements `dragStart` et `dragComplete`. Il doit impérativement être interactif.
clipboard	Objet Clipboard contenant les données à transférer. L'objet Clipboard est référencé dans l'objet NativeDragEvent distribué lors de la séquence glisser-déposer.
dragImage	(Facultatif) Objet BitmapData à afficher lors du glissement. L'image peut stipuler une valeur `alpha`. (Remarque : Microsoft Windows applique systématiquement un fondu alpha fixe aux images glissées.)
offset	(Facultatif) Objet Point qui stipule le décalage de l'image glissée par rapport à la zone sensible de la souris. Utilisez des coordonnées négatives pour déplacer l'image vers le haut et la gauche par rapport au curseur de la souris. Si vous ne définissez pas de décalage, l'angle gauche de l'image glissée est placé sur la zone sensible de la souris.
actionsAllowed	(Facultatif) Objet NativeDragOptions qui indique les actions (copie, déplacement ou liaison) gérées par l'opération de glissement. Si vous n'indiquez aucun argument, toutes les actions sont autorisées. L'objet DragOptions est référencé dans les objets NativeDragEvent pour permettre à la cible potentielle d'un glissement de vérifier si les actions autorisées sont compatibles avec l'objectif du composant cible. Par exemple, un composant « trash » est susceptible de n'accepter que les mouvements de glissement qui autorisent l'action de déplacement.

L'exemple suivant illustre le démarrage d'une opération de glissement d'un objet bitmap chargé à partir d'un fichier. L'exemple charge une image et, lors d'un événement `mouseDown`, démarre l'opération de glissement.

```
 package
{
import flash.desktop.NativeDragManager;
import mx.core.UIComponent;
import flash.display.Sprite;
import flash.display.Loader;
import flash.system.LoaderContext;
import flash.net.URLRequest;
import flash.geom.Point;
import flash.desktop.Clipboard;
import flash.display.Bitmap;
import flash.filesystem.File;
import flash.events.Event;
import flash.events.MouseEvent;

public class DragOutExample extends UIComponent Sprite {
    protected var fileURL:String = "app:/image.jpg";
    protected var display:Bitmap;

    private function init():void {
        loadImage();
    }
    private function onMouseDown(event:MouseEvent):void {
        var bitmapFile:File = new File(fileURL);
        var transferObject:Clipboard = createClipboard(display, bitmapFile);
        NativeDragManager.doDrag(this,
                            transferObject,
                            display.bitmapData,
                            new Point(-mouseX,-mouseY));
    }
    public function createClipboard(image:Bitmap, sourceFile:File):Clipboard {
        var transfer:Clipboard = new Clipboard();
        transfer.setData("bitmap",
                            image,
                            true);
                            // ActionScript 3 Bitmap object by value and by reference
        transfer.setData(ClipboardFormats.BITMAP_FORMAT,
                            image.bitmapData,
                            false);
                            // Standard BitmapData format
        transfer.setData(ClipboardFormats.FILE_LIST_FORMAT,
```

```
                        new Array(sourceFile),
                        false);
                        // Standard file list format
        return transfer;
    }
    private function loadImage():void {
        var url:URLRequest = new URLRequest(fileURL);
        var loader:Loader = new Loader();
        loader.load(url,new LoaderContext());
        loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoadComplete);
    }
    private function onLoadComplete(event:Event):void {
        display = event.target.loader.content;
        var flexWrapper:UIComponent = new UIComponent();
        flexWrapper.addChild(event.target.loader.content);
        addChild(flexWrapper);
        flexWrapper.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
    }
}
}
```

Achèvement d'un transfert par glissement vers l'extérieur

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un utilisateur dépose l'élément glissé en relâchant la souris, l'objet initiateur distribue un événement `nativeDragComplete`. Vous pouvez vérifier la propriété `dropAction` de l'objet événement, puis exécuter l'action appropriée. Par exemple, si l'action correspond à `NativeDragAction.MOVE`, , vous pourriez supprimer l'élément source de son emplacement d'origine. L'utilisateur peut abandonner un mouvement de glissement en relâchant le bouton de la souris lorsque le curseur figure en dehors d'une cible de dépôt appropriée. Le gestionnaire de glissement définit la propriété `dropAction` d'un mouvement abandonné sur `NativeDragAction.NONE`.

Prise en charge du mouvement de glissement vers l'intérieur

Adobe AIR 1.0 et les versions ultérieures

Pour prendre en charge le mouvement de glissement vers l'intérieur, votre application (ou, le plus souvent, un composant visuel de cette dernière) doit répondre aux événements `nativeDragEnter` ou `nativeDragOver`.

Etapes d'une opération de dépôt standard

Adobe AIR 1.0 et les versions ultérieures

La séquence d'événements suivante est caractéristique d'une opération de dépôt :

1. L'utilisateur fait glisser un objet Clipboard vers un composant.
2. Le composant distribue un événement `nativeDragEnter`.

3 Le gestionnaire d'événement `nativeDragEnter` examine l'objet événement pour vérifier les formats de données disponibles et les actions autorisées. Si le composant peut gérer le dépôt, il appelle `NativeDragManager.acceptDragDrop()`.

4 NativeDragManager modifie le curseur de la souris pour indiquer que l'objet peut être déposé.

5 L'utilisateur dépose l'objet sur le composant.

6 Le composant cible distribue un événement `nativeDragDrop`.

7 Le composant cible lit les données au format approprié à partir de l'objet Clipboard au sein de l'objet événement.

8 Si le mouvement de glissement a débuté dans une application AIR, l'objet initiateur interactif distribue un événement `nativeDragComplete`. Si le mouvement a débuté en dehors d'AIR, aucun compte rendu n'est envoyé.

Confirmation d'un mouvement de glissement vers l'intérieur

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un utilisateur fait glisser un élément Clipboard vers les limites d'un composant visuel, celui-ci distribue les événements `nativeDragEnter` et `nativeDragOver`. Pour déterminer si le composant peut accepter l'élément Clipboard, les gestionnaires de ces événements peuvent vérifier les propriétés `clipboard` et `allowedActions` de l'objet événement. Pour signaler que le composant peut accepter le dépôt, le gestionnaire d'événement doit appeler la méthode `NativeDragManager.acceptDragDrop()` et transmettre une référence au composant récepteur. Si plusieurs écouteurs d'événement enregistrés appellent la méthode `acceptDragDrop()`, le dernier gestionnaire de la liste prime. L'appel `acceptDragDrop()` demeure valide jusqu'à ce que la souris quitte les limites de l'objet qui accepte l'élément, déclenchant ainsi l'événement `nativeDragExit`.

Si plusieurs actions sont autorisées par le paramètre `allowedActions` transmis à `doDrag()`, l'utilisateur peut indiquer l'action autorisée qu'il souhaite exécuter en maintenant enfoncée une touche de modification. Le gestionnaire de glissement modifie l'image associée au curseur pour indiquer à l'utilisateur l'action qui se produirait s'il achevait le dépôt. L'action prévue est signalée par la propriété `dropAction` de l'objet NativeDragEvent. L'action associée à un mouvement de glissement est définie à titre indicatif uniquement. Les composants impliqués dans le transfert doivent mettre en œuvre le comportement approprié. Pour achever une action de déplacement, par exemple, l'initiateur du glissement peut supprimer l'élément glissé et la cible du dépôt peut l'ajouter.

La cible du glissement peut limiter l'action de dépôt à l'une des trois actions gérées en définissant la propriété `dropAction` de la classe NativeDragManager. Si un utilisateur tente de sélectionner une autre action par le biais du clavier, NativeDragManager affiche le curseur *unavailable*. Définissez la propriété `dropAction` des gestionnaires associés aux événements `nativeDragEnter` et `nativeDragOver`.

L'exemple suivant illustre un gestionnaire associé à l'événement `nativeDragEnter` ou `nativeDragOver`. Ce gestionnaire accepte un mouvement de glissement vers l'intérieur sous réserve que l'objet Clipboard en cours de glissement contienne des données au format texte.

```
import flash.desktop.NativeDragManager;
import flash.events.NativeDragEvent;

public function onDragIn(event:NativeDragEvent):void{
    NativeDragManager.dropAction = NativeDragActions.MOVE;
    if(event.clipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
        NativeDragManager.acceptDragDrop(this); //'this' is the receiving component
    }
}
```

Achèvement du dépôt

Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur dépose un élément glissé sur un objet interactif qui a accepté le mouvement, ce dernier distribue un événement `nativeDragDrop`. Le gestionnaire de cet événement peut extraire les données de la propriété `clipboard` de l'objet événement.

Si la propriété clipboard contient un format défini par l'application, le paramètre `transferMode` transmis à la méthode `getData()` de l'objet Clipboard détermine si le gestionnaire de glissement renvoie une référence ou une version sérialisée de l'objet.

L'exemple suivant illustre un gestionnaire associé à l'événement `nativeDragDrop` :

```
 import flash.desktop.Clipboard;
import flash.events.NativeDragEvent;

public function onDrop(event:NativeDragEvent):void {
    if (event.clipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)) {
    var text:String =
        String(event.clipboard.getData(ClipboardFormats.TEXT_FORMAT,
                                       ClipboardTransferMode.ORIGINAL_PREFERRED));
}
```

Lorsque le gestionnaire d'événement se referme, l'objet Clipboard n'est plus valide. Toute tentative d'accès à l'objet ou aux données correspondantes génère une erreur.

Mise à jour de l'apparence visuelle d'un composant

Adobe AIR 1.0 et les versions ultérieures

Un composant peut mettre à jour son apparence visuelle en fonction des événements NativeDragEvent. Le tableau ci-dessous décrit les types de modifications effectuées par un composant standard en réponse à divers événements :

Evénement	Description
nativeDragStart	L'objet interactif initiateur peut utiliser l'événement `nativeDragStart` pour indiquer visuellement qu'il est la source du mouvement de glissement.
nativeDragUpdate	L'objet interactif initiateur peut utiliser l'événement nativeDragUpdate pour mettre à jour son état au cours du mouvement. (Cet événement n'existe pas dans AIR pour Linux.)
nativeDragEnter	Un objet interactif récepteur potentiel peut utiliser cet événement pour prendre le focus ou indiquer visuellement s'il peut ou non accepter le dépôt.
nativeDragOver	Un objet interactif récepteur potentiel peut utiliser cet événement pour répondre au mouvement de la souris au sein de l'objet interactif (lorsque la souris pénètre dans une zone « sensible » d'un composant complexe tel qu'un plan de rues, par exemple).
nativeDragExit	Un objet interactif récepteur potentiel peut utiliser cet événement pour restaurer son état lorsqu'un mouvement de glissement quitte ses limites.
nativeDragComplete	L'objet interactif initiateur peut utiliser cet événement pour mettre à jour le modèle de données correspondant (en supprimant un élément d'une liste, par exemple) et restaurer son état visuel.

Suivi de la position de la souris lors d'un mouvement de glissement vers l'intérieur

Adobe AIR 1.0 et les versions ultérieures

Si un mouvement de glissement survole un composant, ce dernier distribue des événements `nativeDragOver`. Ces événements sont distribués toutes les quelques millisecondes, ainsi qu'à chaque déplacement de la souris. L'objet événement `nativeDragOver` permet également de déterminer la position de la souris au-dessus du composant. Connaître la position de la souris peut s'avérer utile dans des circonstances où le composant récepteur est complexe, mais ne possède pas de sous-composants. Par exemple, si votre application a affiché une image bitmap contenant une carte de rue et que vous souhaitez mettre en évidence les zones de la carte dans lesquelles l'utilisateur a fait glisser des informations, vous pouvez utiliser les coordonnées de la souris indiquées par l'événement `nativeDragOver` pour assurer le suivi de la position de la souris au sein de la carte.

Opération glisser-déposer dans HTML

Adobe AIR 1.0 et les versions ultérieures

Pour faire glisser des données vers une application HTML ou hors de cette dernière (ou vers le code HTML affiché dans un objet HTMLLoader et hors de ce dernier), vous disposez des événements glisser-déposer HTML. L'API glisser-déposer HTML vous permet de faire glisser des données vers des éléments DOM ou à partir de ces derniers dans le contenu HTML.

Remarque : *vous pouvez également utiliser les API NativeDragEvent et NativeDragManager d'AIR en écoutant les événements sur l'objet HTMLLoader qui comporte le contenu HTML. L'API HTML est toutefois mieux intégrée au DOM HTML et permet de contrôler le comportement par défaut.*

Comportement par défaut des mouvements glisser-déposer

Adobe AIR 1.0 et les versions ultérieures

L'environnement HTML définit le comportement par défaut des actions glisser-déposer qui impliquent du texte, des images et des URL. Le comportement par défaut permet de faire glisser ces types de données hors d'un élément. Vous ne pouvez toutefois faire glisser que du texte vers un élément, sous réserve qu'il réside dans une zone modifiable de page. Lorsque vous faites glisser du texte entre des zones modifiables d'une page ou au sein de l'une de ces zones, le comportement par défaut consiste à exécuter une action de déplacement. Lorsque vous faites glisser du texte vers une zone modifiable à partir d'une zone non modifiable ou de l'extérieur de l'application, le comportement par défaut consiste à exécuter une action de copie.

Vous pouvez remplacer le comportement par défaut en gérant vous-même les événements glisser-déposer. Pour annuler le comportement par défaut, vous devez appeler les méthodes `preventDefault()` des objets distribués suite aux événements glisser-déposer. Vous pouvez ensuite insérer des données dans la cible du dépôt et supprimer les données de la source du glissement pour exécuter l'action sélectionnée.

Par défaut, l'utilisateur peut sélectionner et faire glisser n'importe quel texte. Il peut faire glisser des images et des liens. La propriété CSS WebKit, `-webkit-user-select`, permet de contrôler le mode de sélection de tout élément HTML. Par exemple, si vous définissez `-webkit-user-select` sur `none`, il est impossible de sélectionner les contenus d'élément et, par conséquent, de les faire glisser. La propriété CSS `-webkit-user-drag` permet également de déterminer s'il est possible de faire glisser un élément global. Le contenu de l'élément est toutefois traité séparément. L'utilisateur peut néanmoins faire glisser une section sélectionnée de texte. Pour plus d'informations, voir « CSS dans AIR » à la page 1015.

Evénements glisser-déposer dans HTML

Adobe AIR 1.0 et les versions ultérieures

Les événements distribués par l'élément initiateur à l'origine d'un glissement sont indiqués dans le tableau suivant :

Evénement	Description
dragstart	Distribué lorsque l'utilisateur démarre l'action de glissement. Le gestionnaire de cet événement peut, le cas échéant, interdire le glissement en appelant la méthode preventDefault() de l'objet événement. Pour déterminer si les données glissées peuvent être copiées, liées ou déplacées, définissez la propriété effectAllowed. Le texte, les images et les liens sélectionnés sont placés dans le Presse-papiers par défaut, mais vous pouvez définir d'autres données pour le mouvement de glissement par le biais de la propriété dataTransfer de l'objet événement.
drag	Distribué continuellement pendant le mouvement de glissement.
dragend	Distribué lorsque l'utilisateur relâche le bouton de la souris pour achever l'action de glissement.

Les événements distribués par une cible de glissement sont les suivants :

Evénement	Description
dragover	Distribué continuellement tant que le mouvement de glissement ne dépasse pas les limites de l'élément. Le gestionnaire de cet événement doit définir la propriété dataTransfer.dropEffect pour indiquer si le dépôt entraîne une action de copie, de déplacement ou de lien lorsque l'utilisateur relâche la souris.
dragenter	Distribué lorsque le mouvement de glissement pénètre dans les limites de l'élément. Si vous modifiez toute propriété d'un objet dataTransfer dans un gestionnaire d'événement dragenter, ces modifications sont rapidement annulées par l'événement dragover suivant. En revanche, il se produit un bref délai entre un événement dragenter et le premier événement dragover susceptible d'entraîner le clignotement du curseur si d'autres propriétés sont définies. Dans de nombreux cas de figure, vous pouvez utiliser le même gestionnaire pour les deux événements.
dragleave	Distribué lorsque le mouvement de glissement quitte les limites de l'élément.
drop	Distribué lorsque l'utilisateur dépose les données sur l'élément. Seul le gestionnaire de l'événement peut accéder aux données en cours de glissement.

L'objet événement distribué en réponse à ces événements est similaire à un événement souris. Les propriétés d'événement souris telles que (`clientX`, `clientY`) et (`screenX`, `screenY`) permettent de déterminer la position de la souris.

La principale propriété d'un objet événement drag correspond à `dataTransfer`, qui contient les données en cours de glissement. L'objet `dataTransfer` en tant que tel dispose des propriétés et méthodes suivantes :

Propriété ou méthode	Description
effectAllowed	Effet autorisé par la source du glissement. Le gestionnaire de l'événement dragstart définit généralement cette valeur. Pour plus d'informations, voir la section « Effets de glissement dans HTML » à la page 640.
dropEffect	Effet sélectionné par la cible ou l'utilisateur. Si vous définissez la propriété `dropEffect` dans un gestionnaire d'événement `dragover` ou `dragenter`, AIR met à jour le curseur de la souris pour indiquer l'effet qui se produit lorsque l'utilisateur relâche le bouton de la souris. Si la propriété `dropEffect` définie ne correspond pas à l'un des effets autorisés, aucun dépôt n'est autorisé et le curseur *unavailable* s'affiche. Si vous n'avez pas défini de propriété `dropEffect` en réponse à l'événement `dragover` ou `dragenter` le plus récent, l'utilisateur peut choisir l'un des effets autorisés par le biais des touches de modification standard gérées par le système d'exploitation. Le dernier effet est signalé par la propriété `dropEffect` de l'objet distribué pour `dragend`. Si l'utilisateur abandonne le dépôt en relâchant le bouton de la souris hors d'une cible appropriée, la propriété `dropEffect` est définie sur `none`.
types	Tableau contenant les chaînes de type MIME associées à chaque format de données que contient l'objet `dataTransfer`.
getData(mimeType)	Extrait les données au format stipulé par le paramètre `mimeType`. La méthode `getData()` ne peut être appelée qu'en réponse à l'événement `drop`.
setData(mimeType)	Ajoute des données à l'objet `dataTransfer` au format stipulé par le paramètre mimeType. Vous pouvez ajouter des données dans divers formats en appelant la méthode `setData()` pour chaque type MIME. Toute donnée placée dans l'objet `dataTransfer` par le comportement de glissement par défaut est effacée. La méthode `setData()` ne peut être appelée qu'en réponse à l'événement `dragstart`.
clearData(mimeType)	Efface toute donnée au format stipulé par le paramètre `mimeType`.
setDragImage(image, offsetX, offsetY)	Définit une image de glissement personnalisée. Vous ne pouvez appeler la méthode `setDragImage()` qu'en réponse à l'événement dragstart et uniquement lorsqu'un élément HTML entier fait l'objet d'un glissement en définissant le style CSS `-webkit-user-drag` sur `element`. Le paramètre `image` peut correspondre à un objet JavaScript Element ou Image.

Types MIME associés à l'événement glisser-déposer HTML

Adobe AIR 1.0 et les versions ultérieures

Les types MIME à utiliser avec l'objet `dataTransfer` d'un événement glisser-déposer HTML sont indiqués dans le tableau suivant :

Format de données	Type MIME
Texte	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Image bitmap	"image/x-vnd.adobe.air.bitmap"
Liste de fichiers	"application/x-vnd.adobe.air.file-list"

Vous disposez également d'autres chaînes MIME, notamment les chaînes définies par une application. D'autres applications risquent toutefois de ne pas pouvoir reconnaître ou utiliser les données transférées. Il vous incombe d'ajouter des données à l'objet `dataTransfer` dans le format attendu.

Important : *seul le code qui s'exécute dans le sandbox d'application peut accéder aux fichiers déposés. Toute tentative de lecture ou de définition de propriété d'un objet File au sein d'un sandbox hors application génère une erreur de sécurité. Pour plus d'informations, voir la section « Gestion des dépôts de fichier dans un sandbox HTML hors application » à la page 644.*

Effets de glissement dans HTML

Adobe AIR 1.0 et les versions ultérieures

L'initiateur du mouvement de glissement peut limiter les effets de glissement autorisés en définissant la propriété `dataTransfer.effectAllowed` du gestionnaire de l'événement `dragstart`. Les valeurs de chaîne suivantes sont prises en charge :

Valeur de chaîne	Description
"none"	Aucune opération de glissement n'est autorisée.
"copy"	Les données sont copiées sur la cible, sans modifier l'original.
"link"	Les données sont partagées avec la cible du dépôt par le biais d'un lien associé à l'original.
"move"	Les données sont copiées sur la cible et supprimées de l'emplacement d'origine.
"copyLink"	Les données peuvent être copiées ou liées.
"copyMove"	Les données peuvent être copiées ou déplacées.
"linkMove"	Les données peuvent être liées ou déplacées.
"all"	Les données peuvent être copiées, déplacées ou liées. *All* correspond à l'effet par défaut lorsque vous interdisez le comportement par défaut.

La cible du mouvement de glissement peut définir la propriété `dataTransfer.dropEffect` pour indiquer l'action exécutée si l'utilisateur achève le dépôt. Si l'effet de dépôt fait partie des actions autorisées, le système affiche le curseur de copie, déplacement ou lien approprié. Dans le cas contraire, le système affiche le curseur *unavailable*. Si aucun effet de dépôt n'est défini par la cible, l'utilisateur peut choisir les actions autorisées par le biais des touches de modification.

Le code suivant définit la valeur `dropEffect` dans les gestionnaires des événements `dragover` et `dragenter` :

```
function doDragStart(event) {
    event.dataTransfer.setData("text/plain","Text to drag");
    event.dataTransfer.effectAllowed = "copyMove";
}

function doDragOver(event) {
    event.dataTransfer.dropEffect = "copy";
}

function doDragEnter(event) {
    event.dataTransfer.dropEffect = "copy";
}
```

Remarque : *bien qu'il soit recommandé de définir systématiquement la propriété `dropEffect` dans le gestionnaire de l'événement `dragenter`, notez que l'événement `dragover` suivant réinitialise la valeur par défaut de la propriété. Définissez `dropEffect` en réponse aux deux événements.*

Glissement des données hors d'un élément HTML

Adobe AIR 1.0 et les versions ultérieures

Le comportement par défaut permet de copier par glissement la plupart des contenus d'une page HTML. Vous pouvez contrôler le contenu autorisé à faire l'objet d'un glissement à l'aide des propriétés CSS `-webkit-user-select` et `-webkit-user-drag`.

Remplacez le comportement glisser-déposer par défaut dans le gestionnaire de l'événement `dragstart`. Appelez la méthode `setData()` de la propriété `dataTransfer` de l'objet événement pour associer vos propres données au mouvement de glissement.

Pour indiquer les effets de glissement pris en charge par un objet source lorsque vous ne vous fondez pas sur le comportement par défaut, définissez la propriété `dataTransfer.effectAllowed` de l'objet distribué pour l'événement `dragstart`. Vous disposez de n'importe quelle combinaison d'effets. Par exemple, si un élément source prend en charge les effets *copy* et *link*, définissez la propriété sur `copyLink`.

Définition des données glissées

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Ajoutez les données associées au mouvement de glissement dans le gestionnaire de l'événement `dragstart` par le biais de la propriété `dataTransfer`. La méthode `dataTransfer.setData()` permet de placer les données dans le Presse-papiers en transmettant le type MIME et les données à transférer.

Par exemple, si votre application contient un élément image doté de l'identifiant *imageOfGeorge*, vous pouvez utiliser le gestionnaire d'événement dragstart suivant. Cet exemple ajoute des représentations d'une photo de George en plusieurs formats de données, augmentant ainsi les chances d'utilisation des données glissées par d'autres applications.

```
function dragStartHandler(event){
    event.dataTransfer.effectAllowed = "copy";

    var dragImage = document.getElementById("imageOfGeorge");
    var dragFile = new air.File(dragImage.src);
    event.dataTransfer.setData("text/plain","A picture of George");
    event.dataTransfer.setData("image/x-vnd.adobe.air.bitmap", dragImage);
    event.dataTransfer.setData("application/x-vnd.adobe.air.file-list",
                               new Array(dragFile));
}
```

Remarque : *lorsque vous appelez la méthode `setData()` de l'objet `dataTransfer`, aucune donnée n'est ajoutée par le comportement glisser-déposer par défaut.*

Glissement de données dans un élément HTML

Adobe AIR 1.0 et les versions ultérieures

Le comportement par défaut n'autorise que le glissement de texte vers des zones modifiables de la page. Vous pouvez indiquer qu'un élément et ses enfants peuvent devenir modifiables en incluant l'attribut `contenteditable` dans la balise de début de l'élément. Vous pouvez également rendre un document entier modifiable en définissant la propriété `designMode` de l'objet Document sur `on`.

Vous pouvez prendre en charge d'autres comportements de glissement vers l'intérieur en gérant les événements `dragenter`, `dragover` et `drop` de tout élément capable d'accepter les données glissées.

Activation du glissement vers l'intérieur

Adobe AIR 1.0 et les versions ultérieures

Pour gérer le mouvement de glissement vers l'intérieur, vous devez au préalable annuler le comportement par défaut. Ecoutez les événements `dragenter` et `dragover` sur tout élément HTML à utiliser à titre de cible de dépôt. Dans les gestionnaires de ces événements, appelez la méthode `preventDefault()` de l'objet événement distribué. Annuler le comportement par défaut permet aux zones non modifiables de recevoir un dépôt.

Obtention des données déposées

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez accéder aux données déposées dans le gestionnaire de l'événement `ondrop` :

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

Utilisez la méthode `dataTransfer.getData()` pour lire les données dans le Presse-papiers en transmettant le type MIME du format de données à lire. Pour identifier les formats de données disponibles, utilisez la propriété `types` de l'objet `dataTransfer`. Le tableau `types` contient la chaîne de type MIME de chaque format disponible.

Lorsque vous annulez le comportement par défaut pour les événements dragenter ou dragover, il vous incombe d'insérer toute donnée déposée à l'emplacement approprié dans le document. Il n'existe pas d'API permettant de convertir une position souris en point d'insertion au sein d'un élément. Cette restriction risque de compliquer la mise en œuvre des mouvements de glissement de type insertion.

Exemple : Annulation du comportement de glissement vers l'intérieur HTML par défaut

Adobe AIR 1.0 et les versions ultérieures

Cet exemple met en œuvre une cible de dépôt qui affiche un tableau contenant tous les formats de données disponibles dans l'élément déposé.

Le comportement par défaut permet d'autoriser le glissement du texte, des liens et des images au sein de l'application. L'exemple remplace le comportement de glissement vers l'intérieur par défaut de l'élément div faisant office de cible du dépôt. La principale étape du processus d'activation du contenu non modifiable de sorte à accepter un mouvement de glissement vers l'intérieur consiste à appeler la méthode `preventDefault()` de l'objet événement distribué pour les événements `dragenter` et `dragover`. En réponse à un événement `drop`, le gestionnaire convertit les données transférées en élément row HTML et insère la ligne dans un tableau pour l'afficher.

```
 <html>
<head>
<title>Drag-and-drop</title>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>
<script language="javascript">
    function init(){
        var target = document.getElementById('target');
        target.addEventListener("dragenter", dragEnterOverHandler);
        target.addEventListener("dragover", dragEnterOverHandler);
        target.addEventListener("drop", dropHandler);

        var source = document.getElementById('source');
        source.addEventListener("dragstart", dragStartHandler);
        source.addEventListener("dragend", dragEndHandler);

        emptyRow = document.getElementById("emptyTargetRow");
    }

    function dragStartHandler(event){
        event.dataTransfer.effectAllowed = "copy";
    }

    function dragEndHandler(event){
        air.trace(event.type + ": " + event.dataTransfer.dropEffect);
    }

    function dragEnterOverHandler(event){
        event.preventDefault();
    }

    var emptyRow;
    function dropHandler(event){
        for(var prop in event){
            air.trace(prop + " = " + event[prop]);
        }
        var row = document.createElement('tr');
        row.innerHTML = "<td>" + event.dataTransfer.getData("text/plain") + "</td>" +
            "<td>" + event.dataTransfer.getData("text/html") + "</td>" +
            "<td>" + event.dataTransfer.getData("text/uri-list") + "</td>" +
            "<td>" + event.dataTransfer.getData("application/x-vnd.adobe.air.file-list") +
            "</td>";

        var imageCell = document.createElement('td');
        if((event.dataTransfer.types.toString()).search("image/x-vnd.adobe.air.bitmap") > -
1){
            imageCell.appendChild(event.dataTransfer.getData("image/x-
vnd.adobe.air.bitmap"));
        }
        row.appendChild(imageCell);
        var parent = emptyRow.parentNode;
        parent.insertBefore(row, emptyRow);
    }
</script>
</head>
<body onLoad="init()" style="padding:5px">
<div>
    <h1>Source</h1>
```

```
       <p>Items to drag:</p>
    <ul id="source">
        <li>Plain text.</li>
           <li>HTML <b>formatted</b> text.</li>
           <li>A <a href="http://www.adobe.com">URL.</a></li>
           <li><img src="icons/AIRApp_16.png" alt="An image"/></li>
           <li style="-webkit-user-drag:none;">
        Uses "-webkit-user-drag:none" style.
           </li>
           <li style="-webkit-user-select:none;">
        Uses "-webkit-user-select:none" style.
           </li>

       </ul>
</div>
<div id="target" style="border-style:dashed;">
    <h1 >Target</h1>
       <p>Drag items from the source list (or elsewhere).</p>
       <table id="displayTable" border="1">
        <tr><th>Plain text</th><th>Html text</th><th>URL</th><th>File list</th><th>Bitmap
Data</th></tr>
           <tr
id="emptyTargetRow"><td> </td><td> </td><td> </td><td> </td><td> </
td></tr>
       </table>
       </div>
</div>
</body>
</html>
```

Gestion des dépôts de fichier dans un sandbox HTML hors application

Adobe AIR 1.0 et les versions ultérieures

Un contenu hors application ne peut pas accepter les objets File qui résultent d'un glissement des fichiers vers une application AIR. Il est également impossible de transmettre l'un de ces objets File à un contenu d'application via un pont de sandbox. (Il est nécessaire d'accéder aux propriétés d'objet pendant la sérialisation.) Vous pouvez toutefois continuer à déposer des fichiers dans votre application en écoutant les événements nativeDragDrop AIR de l'objet HTMLLoader.

En temps normal, si un utilisateur dépose un fichier dans un cadre qui héberge un contenu hors application, l'événement dépôt n'est pas propagé de l'enfant au parent. Toutefois, puisque les événements distribués par l'objet HTMLLoader (qui comporte tous les contenus HTML d'une application AIR) ne sont pas intégrés au flux d'événement HTML, vous pouvez continuer à recevoir l'événement dépôt dans un contenu d'application.

Pour recevoir l'événement associé à un dépôt de fichier, le document parent ajoute un écouteur d'événement à l'objet HTMLLoader par le biais de la référence fournie par `window.htmlLoader` :

```
window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

L'exemple suivant utilise un document parent qui charge une page enfant dans un sandbox distant (http://localhost/). Le parent écoute l'événement `nativeDragDrop` sur l'objet HTMLLoader et suit en sortie le modèle d'URL file.

```
 <html>
<head>
<title>Drag-and-drop in a remote sandbox</title>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>
<script language="javascript">
    window.htmlLoader.addEventListener("nativeDragDrop",function(event){
        var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
        air.trace(filelist[0].url);
    });
</script>
</head>
<body>
       <iframe src="child.html"
               sandboxRoot="http://localhost/"
               documentRoot="app:/"
               frameBorder="0"  width="100%" height="100%">
       </iframe>
</body>
</html>
```

Le document enfant doit présenter une cible de dépôt valide en appelant la méthode `preventDefault()` de l'objet Event dans les gestionnaires d'événement HTML `dragenter` et `dragover`, sous peine que l'événement dépôt ne se produise jamais.

```
 <html>
<head>
    <title>Drag and drop target</title>
    <script language="javascript" type="text/javascript">
        function preventDefault(event){
            event.preventDefault();
        }
    </script>
</head>
<body ondragenter="preventDefault(event)" ondragover="preventDefault(event)">
<div>
<h1>Drop Files Here</h1>
</div>
</body>
</html>
```

Dépôt de fichiers promis

Adobe AIR 2 et ultérieur

Un fichier promis est un format de Presse-papiers de type glisser-déposer grâce auquel un utilisateur peut faire glisser un fichier qui n'existe pas encore hors d'une application AIR. Par le biais de fichiers promis, une application pourrait, par exemple, autoriser un utilisateur à faire glisser une icône de proxy vers un dossier du bureau. L'icône de proxy représente un fichier ou des données dont la disponibilité est connue à une URL donnée. Lorsque l'utilisateur dépose l'icône, le moteur d'exécution télécharge les données et écrit le fichier à l'emplacement de dépôt.

La classe URLFilePromise d'une application AIR permet de glisser-déposer des fichiers auxquels il est possible d'accéder à partir d'une URL. L'implémentation d'URLFilePromise figure dans la bibliothèque principale d'AIR et fait partie du kit de développement d'AIR 2. Utilisez le fichier aircore.swc ou le fichier aircore.swf qui réside dans le répertoire SDK frameworks/libs/air.

Vous pouvez également implémenter votre propre logique de fichier promis par le biais de l'interface IFilePromise (définie dans le package flash.desktop du moteur d'exécution).

D'un point de vue conceptuel, les fichiers promis sont similaires à un rendu différé qui fait appel à une fonction de gestion des données du Presse-papiers. Utilisez les fichiers promis au lieu d'un rendu différé lorsque vous glissez-déposez des fichiers. La technique de rendu différé entraîne parfois des pauses indésirables du mouvement de glissement lors de la génération ou du téléchargement des données. Faites appel au rendu différé pour les opérations de copie et de collage, qui ne prennent pas en charge les fichiers promis.

Restrictions associées à l'utilisation de fichiers promis

Par rapport aux autres formats de données gérés par un Presse-papiers de type glisser-déposer, les fichiers promis sont soumis aux restrictions suivantes :

- Si les fichiers promis peuvent être transférés à partir d'une application AIR, il est impossible de les déposer dans une application AIR.
- Les fichiers promis ne sont pas pris en charge par tous les systèmes d'exploitation. La propriété `Clipboard.supportsFilePromise` permet de vérifier si les fichiers promis sont pris en charge sur le système hôte. Si un système ne prend pas en charge les fichiers promis, il vous incombe de proposer un autre mécanisme de téléchargement ou de génération des données de fichier.
- Il est impossible d'utiliser les fichiers promis avec le Presse-papiers de type copier et coller (`Clipboard.generalClipboard`).

Voir aussi

flash.desktop.IFilePromise

air.desktop.URLFilePromise

Dépôt de fichiers distants

Adobe AIR 2 et ultérieur

La classe URLFilePromise permet de créer des objets de fichiers promis représentant les fichiers ou données disponibles à partir d'une URL. Ajoutez un ou plusieurs objets de fichiers promis au Presse-papiers en faisant appel au format de Presse-papiers `FILE_PROMISE_LIST`. Dans l'exemple suivant, un fichier unique, disponible à l'adresse http://www.example.com/foo.txt, est téléchargé et enregistré à l'emplacement de dépôt sous le nom bar.txt. (Le nom du fichier distant et celui du fichier local ne sont pas nécessairement identiques.)

```
if( Clipboard.supportsFilePromise )
{
    var filePromise:URLFilePromise = new URLFilePromise();
    filePromise.request = new URLRequest("http://example.com/foo.txt");
    filePromise.relativePath = "bar.txt";

    var fileList:Array = new Array( filePromise );
    var clipboard:Clipboard = new Clipboard();
    clipboard.setData( ClipboardFormats.FILE_PROMISE_LIST_FORMAT, fileList );
    NativeDragManager.doDrag( dragSource, clipboard );
}
```

Vous pouvez autoriser l'utilisateur à faire glisser simultanément plusieurs fichiers en ajoutant d'autres objets de fichiers promis au tableau affecté au Presse-papiers. Vous pouvez également définir des sous-répertoires dans la propriété `relativePath` de sorte que certains ou la totalité des fichiers concernés par l'opération soient placés dans un sous-répertoire relatif à l'emplacement du dépôt.

L'exemple suivant illustre la procédure de lancement d'une opération de dépôt qui inclut plusieurs fichiers promis. Dans cet exemple, une page html, *article.html*, est placée dans le Presse-papiers en tant que fichier promis, accompagnée des deux fichiers d'images liés correspondants. Les images sont copiées dans un sous-dossier *images* afin de conserver les liens relatifs.

```
if( Clipboard.supportsFilePromise )
{   //Create the promise objects
    var filePromise:URLFilePromise = new URLFilePromise();
    filePromise.request = new URLRequest("http://example.com/article.html");
    filePromise.relativePath = "article.html";

    var image1Promise:URLFilePromise = new URLFilePromise();
    image1Promise.request = new URLRequest("http://example.com/images/img_1.jpg");
    image1Promise.relativePath = "images/img_1.html";
    var image2Promise:URLFilePromise = new URLFilePromise();
    image2Promise.request = new URLRequest("http://example.com/images/img_2.jpg");
    image2Promise.relativePath = "images/img_2.jpg";


    //Put the promise objects onto the clipboard inside an array
    var fileList:Array = new Array( filePromise, image1Promise, image2Promise );
    var clipboard:Clipboard = new Clipboard();
    clipboard.setData( ClipboardFormats.FILE_PROMISE_LIST_FORMAT, fileList );
    //Start the drag operation
    NativeDragManager.doDrag( dragSource, clipboard );
}
```

Implémentation de l'interface IFilePromise

Adobe AIR 2 et ultérieur

Pour associer des fichiers promis aux ressources auxquelles il est impossible d'accéder par le biais d'un objet URLFilePromise, vous pouvez implémenter l'interface IFilePromise dans une classe personnalisée. L'interface IFilePromise définit les méthodes et propriétés utilisées par le moteur d'exécution AIR pour accéder aux données à écrire dans un fichier une fois le fichier promis déposé.

Une implémentation d'IFilePromise transmet un autre objet au moteur d'exécution AIR qui fournit les données associées au fichier promis. Cet objet doit implémenter l'interface IDataInput, utilisée par le moteur d'exécution AIR pour lire les données. Par exemple, la classe URLFilePromise, qui implémente IFilePromise, fait appel à un objet URLStream en tant que fournisseur de données.

AIR peut lire les données en mode synchrone et asynchrone. L'implémentation d'IFilePromise indique le mode d'accès pris en charge en renvoyant la valeur correspondante dans la propriété `isAsync`. Si un accès asynchrone aux données est proposé, l'objet fournisseur de données doit implémenter l'interface IEventDispatcher et distribuer les événements requis, tels que `open`, `progress` et `complete`.

Vous pouvez utiliser une classe personnalisée ou l'une des classes intégrées suivantes en tant que fournisseur de données associé à un fichier promis :

- ByteArray (accès synchrone)
- FileStream (accès synchrone ou asynchrone)
- Socket (accès asynchrone)
- URLStream (accès asynchrone)

Pour implémenter l'interface IFilePromise, vous devez fournir le code des fonctions et propriétés suivantes :

- `open():IDataInput` — Renvoie l'objet fournisseur de données à partir duquel sont lues les données associées au fichier promis. L'objet doit implémenter l'interface IDataInput. Si les données sont proposées en mode asynchrone, l'objet doit également implémenter l'interface IEventDispatcher et distribuer les événements requis (voir « Utilisation d'un fournisseur de données asynchrone dans un fichier promis » à la page 650).
- `get relativePath():String` — Fournit le chemin d'accès, nom inclus, du fichier créé. Le chemin est résolu relativement à l'emplacement de dépôt choisi par l'utilisateur dans le cadre de l'opération de glisser-déposer. Pour vous assurer que le chemin utilise le caractère de séparation adapté au système d'exploitation hôte, faites appel à la constante `File.separator` lorsque vous définissez des chemins contenant des répertoires. Vous pouvez ajouter une fonction de définition (setter) ou utiliser un paramètre constructeur pour autoriser la définition du chemin à l'exécution.
- `get isAsync():Boolean` — Indique au moteur d'exécution AIR si l'objet fournisseur de données fournit des données en mode asynchrone ou synchrone.
- `close():void` — Fonction appelée par le moteur d'exécution une fois les données entièrement lues ou si une erreur empêche leur lecture complète. Vous pouvez faire appel à cette fonction pour nettoyer les ressources.
- `reportError( e:ErrorEvent ):void` — Fonction appelée par le moteur d'exécution s'il se produit une erreur lors de la lecture des données.

Toutes les méthodes IFilePromise sont appelées par le moteur d'exécution dans le cadre d'une opération de glisser-déposer qui concerne le fichier promis. En règle générale, la logique de l'application ne devrait pas appeler directement ces méthodes.

Utilisation d'un fournisseur de données synchrone dans un fichier promis

Adobe AIR 2 et ultérieur

La technique d'implémentation de l'interface IFilePromise la plus simple consiste à utiliser un objet fournisseur de données synchrone tel que ByteArray ou un objet FileStream synchrone. Dans l'exemple suivant, un objet ByteArray est créé, rempli de données et renvoyé lorsque la méthode `open()` est appelée.

```
package
{
    import flash.desktop.IFilePromise;
    import flash.events.ErrorEvent;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class SynchronousFilePromise implements IFilePromise
    {
        private const fileSize:int = 5000; //size of file data
        private var filePath:String = "SynchronousFile.txt";

        public function get relativePath():String
        {
            return filePath;
        }

        public function get isAsync():Boolean
        {
            return false;
        }

        public function open():IDataInput
        {
            var fileContents:ByteArray = new ByteArray();

            //Create some arbitrary data for the file
            for( var i:int = 0; i < fileSize; i++ )
            {
                fileContents.writeUTFBytes( 'S' );
            }

            //Important: the ByteArray is read from the current position
            fileContents.position = 0;
            return fileContents;
        }

        public function close():void
        {
            //Nothing needs to be closed in this case.
        }

        public function reportError(e:ErrorEvent):void
        {
            trace("Something went wrong: " + e.errorID + " - " + e.type + ", " + e.text );
        }
    }
}
```

En pratique, les fichiers promis synchrones sont d'une utilité limitée. Si le volume de données est réduit, il est tout aussi simple de créer un fichier dans un répertoire temporaire et d'ajouter un tableau contenant une liste de fichiers standard au Presse-papiers glisser-déposer. En revanche, si le volume de données est élevé ou que la génération de données sollicite d'importantes ressources informatiques, un long processus synchrone s'impose. Les longs processus synchrones risquent de bloquer les mises à jour de l'interface utilisateur pendant une durée prolongée. L'application semble alors non réactive. Pour parer à ce problème, vous pouvez créer un fournisseur de données asynchrone piloté par une horloge.

Utilisation d'un fournisseur de données asynchrone dans un fichier promis

Adobe AIR 2 et ultérieur

Lorsque vous utilisez un objet fournisseur de données asynchrone, la propriété `isAsync` d'IFilePromise doit être définie sur `true` et l'objet renvoyé par la méthode `open()` doit implémenter l'interface IEventDispatcher. Le moteur d'exécution écoute plusieurs événements, permettant ainsi d'utiliser divers objets intégrés en tant que fournisseurs de données. Les événements `progress` sont, par exemple, distribués par les objets FileStream et URLStream, alors que les événements `socketData` sont distribués par les objets Socket. Le moteur d'exécution écoute les événements appropriés qui émanent de tous ces objets.

Les événements suivants étayent le processus de lecture des données à partir de l'objet fournisseur de données :

- Event.OPEN — Indique au moteur d'exécution que la source de données est prête.
- ProgressEvent.PROGRESS — Indique au moteur d'exécution que les données sont disponibles. Le moteur d'exécution lit le volume de données disponibles dans l'objet fournisseur de données.
- ProgressEvent.SOCKET_DATA — Indique au moteur d'exécution que les données sont disponibles. L'événement `socketData` est distribué par les objets basés sur un socket. Pour les autres types d'objets, distribuez un événement `progress`. (Le moteur d'exécution écoute les deux événements pour savoir quand les données peuvent être lues.)
- Event.COMPLETE — Indique au moteur d'exécution que la lecture des données est terminée.
- Event.CLOSE — Indique au moteur d'exécution que la lecture des données est terminée. (A ce titre, le moteur d'exécution écoute les événements `close` et `complete`.)
- IOErrorEvent.IOERROR — Indique au moteur d'exécution qu'il s'est produit une erreur lors de la lecture des données. Le moteur d'exécution abandonne la création de fichier et appelle la méthode `close()` d'IFilePromise.
- SecurityErrorEvent.SECURITY_ERROR — Indique au moteur d'exécution qu'il s'est produit une erreur liée à la sécurité. Le moteur d'exécution abandonne la création de fichier et appelle la méthode `close()` d'IFilePromise.
- HTTPStatusEvent.HTTP_STATUS — Utilisé en conjonction avec `httpResponseStatus` par le moteur d'exécution pour s'assurer que les données disponibles représentent le contenu requis, plutôt qu'un message d'erreur (une page 404, par exemple). Les objets basés sur le protocole HTTP doivent distribuer cet événement.
- HTTPStatusEvent.HTTP_RESPONSE_STATUS — Utilisé en conjonction avec `httpStatus` par le moteur d'exécution pour s'assurer que les données disponibles représentent le contenu requis. Les objets basés sur le protocole HTTP doivent distribuer cet événement.

Le fournisseur de données doit distribuer ces événements dans l'ordre suivant :

1. événement `open`
2. événements `progress` ou `socketData`
3. événement `complete` ou `close`

Remarque : *les objets intégrés, FileStream, Socket et URLStream, distribuent automatiquement les événements appropriés.*

L'exemple suivant crée un fichier promis en faisant appel à un fournisseur de données asynchrone personnalisé. La classe du fournisseur de données constitue une extension de ByteArray (pour la prise en charge d'IDataInput) et implémente l'interface IEventDispatcher. A chaque événement associé à l'horloge, l'objet génère un bloc de données et distribue un événement « progress » pour avertir le moteur d'exécution que les données sont disponibles. Lorsque le volume de données produit est suffisant, l'objet distribue un événement « complete ».

```
package
{
import flash.events.Event;
import flash.events.EventDispatcher;
import flash.events.IEventDispatcher;
import flash.events.ProgressEvent;
import flash.events.TimerEvent;
import flash.utils.ByteArray;
import flash.utils.Timer;

[Event(name="open", type="flash.events.Event.OPEN")]
[Event(name="complete",  type="flash.events.Event.COMPLETE")]
[Event(name="progress", type="flash.events.ProgressEvent")]
[Event(name="ioError", type="flash.events.IOErrorEvent")]
[Event(name="securityError", type="flash.events.SecurityErrorEvent")]
public class AsyncDataProvider extends ByteArray implements IEventDispatcher
{
    private var dispatcher:EventDispatcher = new EventDispatcher();
    public var fileSize:int = 0; //The number of characters in the file
    private const chunkSize:int = 1000; //Amount of data written per event
    private var dispatchDataTimer:Timer = new Timer( 100 );
    private var opened:Boolean = false;

    public function AsyncDataProvider()
    {
        super();
        dispatchDataTimer.addEventListener( TimerEvent.TIMER, generateData );
    }

    public function begin():void{
        dispatchDataTimer.start();
    }

    public function end():void
    {
        dispatchDataTimer.stop();
    }
    private function generateData( event:Event ):void
    {
        if( !opened )
        {
            var open:Event = new Event( Event.OPEN );
            dispatchEvent( open );
            opened = true;
        }
        else if( position + chunkSize < fileSize )
        {
            for( var i:int = 0; i <= chunkSize; i++ )
            {
                writeUTFBytes( 'A' );
            }
            //Set position back to the start of the new data
            this.position -= chunkSize;
            var progress:ProgressEvent =
                new ProgressEvent( ProgressEvent.PROGRESS, false, false, bytesAvailable,
bytesAvailable + chunkSize);
            dispatchEvent( progress )
```

```
        }
        else
        {
            var complete:Event = new Event( Event.COMPLETE );
            dispatchEvent( complete );
        }
    }
     //IEventDispatcher implementation
    public function addEventListener(type:String, listener:Function,
useCapture:Boolean=false, priority:int=0, useWeakReference:Boolean=false):void
    {
        dispatcher.addEventListener( type, listener, useCapture, priority, useWeakReference );
    }

    public function removeEventListener(type:String, listener:Function,
useCapture:Boolean=false):void
    {
        dispatcher.removeEventListener( type, listener, useCapture );
    }

    public function dispatchEvent(event:Event):Boolean
    {
        return dispatcher.dispatchEvent( event );
    }

    public function hasEventListener(type:String):Boolean
    {
        return dispatcher.hasEventListener( type );
    }

    public function willTrigger(type:String):Boolean
    {
        return dispatcher.willTrigger( type );
    }
}
}
```

Remarque : *puisque la classe AsyncDataProvider utilisée dans l'exemple constitue une extension de ByteArray, elle ne peut pas étendre également EventDispatcher. Pour implémenter l'interface IEventDispatcher, la classe fait appel à un objet EventDispatcher interne et transmet à ce dernier les appels de la méthode IEventDispatcher. Vous pouvez également étendre EventDispatcher et implémenter IDataInput (voire implémenter les deux interfaces).*

L'implémentation asynchrone d'IFilePromise est quasiment identique à l'implémentation synchrone, à quelques différences près : `isAsync` renvoie `true` et la méthode `open()` renvoie un objet de données asynchrone :

```
package
{
    import flash.desktop.IFilePromise;
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.utils.IDataInput;

    public class AsynchronousFilePromise extends EventDispatcher implements IFilePromise
    {
        private var fileGenerator:AsyncDataProvider;
        private const fileSize:int = 5000; //size of file data
        private var filePath:String = "AsynchronousFile.txt";

        public function get relativePath():String
        {
            return filePath;
        }

        public function get isAsync():Boolean
        {
            return true;
        }

        public function open():IDataInput
        {
            fileGenerator = new AsyncDataProvider();
            fileGenerator.fileSize = fileSize;
            fileGenerator.begin();
            return fileGenerator;
        }

        public function close():void
        {
            fileGenerator.end();
        }

        public function reportError(e:ErrorEvent):void
        {
            trace("Something went wrong: " + e.errorID + " - " + e.type + ", " + e.text );
        }
    }
}
```

Chapitre 36 : Utilisation des menus

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Faites appel aux classes de l'API de menus contextuels pour modifier le menu contextuel des applications Web Flex et Flash Player.

Les classes de l'API de menus natifs permettent de définir des menus en incrustation, contextuels, de fenêtre et d'application dans Adobe® AIR®.

Principes de base des menus

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour obtenir une explication rapide de la création de menus natifs dans une application AIR et des exemples de code, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Ajout de menus natifs à une application AIR (Flex)
- Ajout de menus natifs à une application AIR (Flash)

Les classes de menus natifs vous permettent d'accéder aux fonctions de menu natif du système d'exploitation sur lequel s'exécute votre application. Des objets NativeMenu peuvent être utilisés pour les menus d'application (sous Mac OS X), les menus de fenêtre (sous Windows et Linux), les menus contextuels et les menus en incrustation.

En dehors d'AIR, vous disposez des classes de menus contextuels pour modifier le menu contextuel automatiquement affiché par Flash Player lorsqu'un utilisateur clique avec le bouton droit de la souris ou en maintenant la touche Commande enfoncée sur un objet de votre application. (Aucun menu contextuel automatique ne s'affiche pour les applications AIR.)

Classes de menus

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les classes de menus sont les suivantes :

Package	Classes
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

Types de menus

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

AIR prend en charge les types de menus suivants :

Menus contextuels Un menu contextuel s'affiche lorsque l'utilisateur clique avec le bouton droit de la souris ou en maintenant la touche Commande enfoncée sur un objet interactif dans du contenu SWF ou sur un élément de document dans du contenu HTML.

Un menu contextuel s'affiche automatiquement dans le moteur d'exécution Flash Player. Vous disposez des classes ContextMenu et ContextMenuItem pour ajouter vos propres commandes au menu. Vous pouvez également supprimer certaines des commandes intégrées, mais pas toutes.

Dans le moteur d'exécution AIR, vous pouvez créer un menu contextuel à l'aide de la classe NativeMenu ou ContextMenu. Un contenu HTML dans AIR vous permet d'utiliser les API Webkit HTML et JavaScript pour ajouter des menus contextuels aux éléments HTML.

Menus d'application (AIR uniquement) Un menu d'application est un menu global qui s'applique à la totalité de l'application. Les menus d'application sont pris en charge par Mac OS X, mais pas par Windows ni Linux. Sous Mac OS x, le système d'exploitation crée automatiquement un menu d'application. Vous pouvez utiliser l'API de menus d'AIR pour ajouter des éléments et des sous-menus aux menus standard. Vous pouvez ajouter des écouteurs pour gérer les commandes de menu existantes et vous pouvez supprimer des éléments existants.

Menus de fenêtre (AIR uniquement) Un menu de fenêtre est associé à une fenêtre unique et s'affiche sous la barre de titre. Pour ajouter des menus à une fenêtre, vous créez un objet NativeMenu et l'affectez à la propriété `menu` de l'objet NativeWindow. Les menus de fenêtre sont pris en charge par les systèmes d'exploitation Windows et Linux, mais pas par Mac OS X. Les menus de fenêtre natifs ne peuvent être utilisés qu'avec les fenêtres disposant d'un chrome système.

Menus d'icônes de la barre d'état système et du Dock (AIR uniquement) Similaires aux menus contextuels, les menus d'icône sont affectés à l'icône d'une application dans le Dock de Mac OS X ou dans la zone de notification de la barre des tâches de Windows et Linux. Les menus d'icônes de la barre d'état système et du Dock utilisent la classe NativeMenu. Sous Mac OS X, les éléments du menu sont ajoutés au-dessus des éléments standard du système d'exploitation. Sous Windows et Linux, le menu standard n'existe pas.

Menus en incrustation (AIR uniquement) Un menu en incrustation AIR est similaire à un menu contextuel mais il n'est pas nécessairement associé à un objet ou composant d'application spécifique. Vous pouvez afficher des menus en incrustation n'importe où dans une fenêtre en appelant la méthode `display()` de tout objet NativeMenu.

Menus personnalisés Les menus natifs sont dessinés entièrement par le système d'exploitation et, en tant que tels, existent en dehors des modèles de rendu Flash et HTML. Au lieu d'utiliser des menus natifs, vous pouvez créer des menus personnalisés non natifs à l'aide de MXML, ActionScript, ou JavaScript (dans AIR). Les menus de ce type doivent s'afficher complètement dans le contenu d'une application.

Menus Flex La structure Adobe® Flex™ propose un ensemble de composants de menu Flex. Les menus Flex sont dessinés par le moteur d'exécution plutôt que par le système d'exploitation et ne sont pas des menus *natifs*. Il est possible d'utiliser un composant de menu Flex pour des fenêtres Flex sans chrome système. Autre avantage, vous pouvez spécifier des menus par déclaration au format MXML. Si vous utilisez la structure Flex, faites appel aux classes de menu Flex pour les menus de fenêtre plutôt qu'aux classes natives.

Menus par défaut (AIR uniquement)

Les menus par défaut suivants sont proposés par le système d'exploitation ou une classe AIR intégrée :

- Menu d'application sous Mac OS X
- Menu d'icône du Dock sous Mac OS X

- Menu contextuel associé au texte et aux images sélectionnés dans du contenu HTML
- Menu contextuel associé au texte sélectionné dans un objet TextField (ou un objet étendant TextField)

Présentation des menus contextuels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans du contenu SWF, il est possible d'associer un menu contextuel à tout objet qui hérite des propriétés ou méthodes de InteractiveObject en affectant un objet menu à sa propriété `contextMenu`. Plusieurs commandes sont incluses par défaut, notamment des commandes d'avance, de recul, d'impression, de qualité et de zoom. Dans le moteur d'exécution AIR, l'objet menu affecté à `contextMenu` peut être de type NativeMenu ou ContextMenu. Dans le moteur d'exécution Flash Player, seule la classe ContextMenu est disponible.

Vous pouvez écouter les événements des menus natifs ou des menus contextuels lorsque vous utilisez les classes ContextMenu et ContextMenuItem ; les deux sont distribués. L'un des avantages des propriétés de l'objet ContextMenuEvent est que contextMenuOwner identifie l'objet auquel le menu est associé et `mouseTarget` identifie l'objet sur lequel l'utilisateur a cliqué pour ouvrir le menu. Ces informations ne sont pas disponibles avec l'objet NativeMenuEvent.

L'exemple suivant crée un sprite et l'ajoute à un menu contextuel edit simple :

```
 var sprite:Sprite = new Sprite();
sprite.contextMenu = createContextMenu()
private function createContextMenu():ContextMenu{
    var editContextMenu:ContextMenu = new ContextMenu();
    var cutItem:ContextMenuItem = new ContextMenuItem("Cut")
    cutItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doCutCommand);
    editContextMenu.customItems.push(cutItem);

    var copyItem:ContextMenuItem = new ContextMenuItem("Copy")
    copyItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doCopyCommand);
    editContextMenu.customItems.push(copyItem);

    var pasteItem:ContextMenuItem = new ContextMenuItem("Paste")
    pasteItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, doPasteCommand);
    editContextMenu.customItems.push(pasteItem);

    return editContextMenu
}
private function doCutCommand(event:ContextMenuEvent):void{trace("cut");}
private function doCopyCommand(event:ContextMenuEvent):void{trace("copy");}
private function doPasteCommand(event:ContextMenuEvent):void{trace("paste");}
```

Remarque : *contrairement au contenu SWF affiché dans un environnement de navigateur, dans AIR, les menus contextuels ne possèdent pas de commandes intégrées.*

Personnalisation d'un menu contextuel Flash Player

Dans un navigateur ou un fichier de projection, les menus contextuels associés à un contenu SWF contiennent systématiquement des éléments intégrés. Vous pouvez supprimer toutes ces commandes par défaut du menu, à l'exception des commandes Paramètres et A propos. Lorsque vous définissez la propriété Stage `showDefaultContextMenu` sur `false`, ces commandes sont supprimées du menu contextuel.

Pour créer un menu contextuel personnalisé pour un objet d'affichage particulier, créez une occurrence de la classe ContextMenu, appelez la méthode `hideBuiltInItems()` et affectez cette occurrence à la propriété `contextMenu` de cette occurrence de DisplayObject. L'exemple suivant crée un carré dessiné dynamiquement avec une commande de menu contextuel qui permet de modifier sa couleur au hasard :

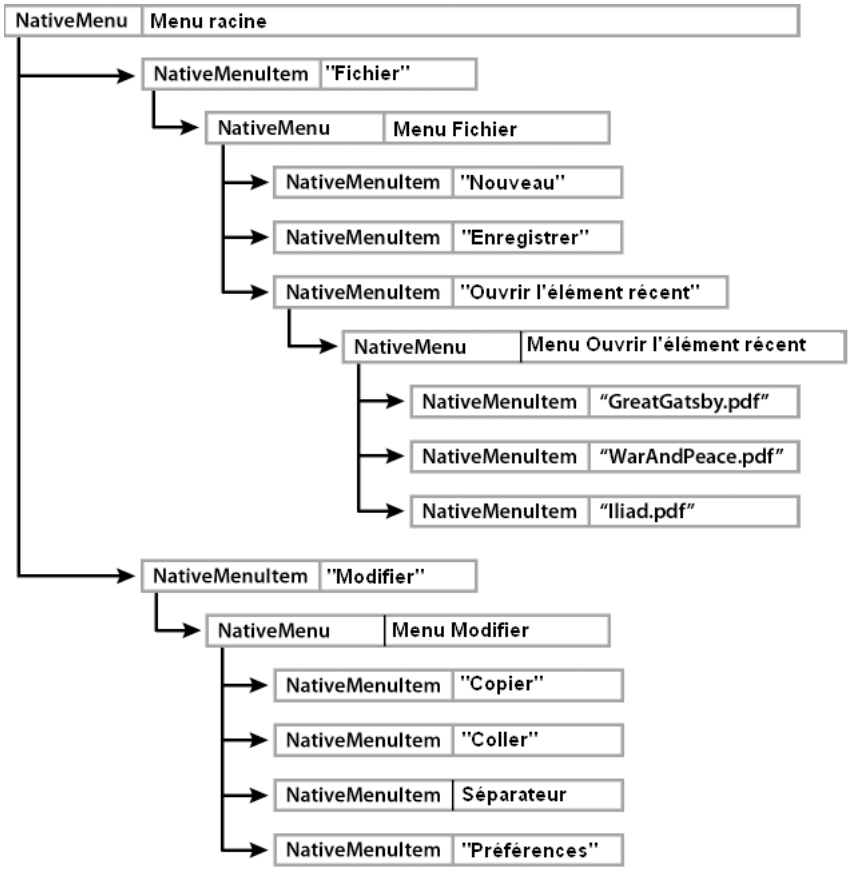
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

var menuItem:ContextMenuItem = new ContextMenuItem("Change Color");
menuItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT,changeColor);
var customContextMenu:ContextMenu = new ContextMenu();
customContextMenu.hideBuiltInItems();
customContextMenu.customItems.push(menuItem);
square.contextMenu = customContextMenu;

function changeColor(event:ContextMenuEvent):void
{
    square.transform.colorTransform = getRandomColor();
}
function getRandomColor():ColorTransform
{
    return new ColorTransform(Math.random(), Math.random(), Math.random(),1,(Math.random() *
512) - 255, (Math.random() * 512) -255, (Math.random() * 512) - 255, 0);
}
```

Structure de menu natif (AIR)

Adobe AIR 1.0 et les versions ultérieures

La structure des menus natifs est par nature hiérarchique. Les objets NativeMenu contiennent des objets enfant NativeMenuItem. Les objets NativeMenuItem qui représentent des sous-menus peuvent eux aussi contenir des objets NativeMenu. L'objet menu du niveau supérieur ou racine de la structure représente la barre de menus des menus d'application et de fenêtre. (Les menus contextuels, d'icônes et en incrustation ne possèdent pas de barre de menus.)

Le schéma suivant illustre la structure d'un menu typique. Le menu racine représente la barre de menus et contient deux éléments de menu référençant les sous-menus *Fichier* et *Modifier*. Le sous-menu Fichier de cette structure contient deux éléments de commande et un élément référençant le sous-menu *Ouvrir l'élément récent* qui, lui-même, contient trois éléments. Le sous-menu Modifier contient trois commandes et un séparateur.



Pour définir un sous-menu, vous devez disposer d'un objet NativeMenu et d'un objet NativeMenuItem. L'objet NativeMenuItem définit le libellé affiché dans le menu parent et permet à l'utilisateur d'ouvrir le sous-menu. L'objet NativeMenu sert de conteneur des éléments du sous-menu. Par le biais de sa propriété `submenu`, l'objet NativeMenuItem référence l'objet NativeMenu.

Vous trouverez un exemple de code qui crée ce menu à la section « Exemple de menu natif : menu de fenêtre et d'application (AIR) » à la page 666.

Evénements de menu

Adobe AIR 1.0 et les versions ultérieures

Les objets NativeMenu et NativeMenuItem distribuent tous deux des événements `preparing`, `displaying`, et `select` :

Preparing : lorsque l'objet est sur le point de démarrer une interaction utilisateur, le menu et les sous-menus correspondants distribuent un événement `preparing` à tout écouteur enregistré. Parmi les interactions figurent l'ouverture du menu et la sélection d'une option par le biais d'un raccourci clavier.

Remarque : *l'événement* `preparing` *n'est disponible que dans Adobe AIR 2.6 et les versions ultérieures.*

Displaying : juste avant son affichage, un menu et ses options distribuent un événement `displaying` à tout écouteur enregistré.

Les événements `preparing` et `displaying` permettent de mettre à jour le contenu du menu ou l'apparence de l'option avant son affichage à l'intention de l'utilisateur. Par exemple, dans l'écouteur de l'événement `displaying` d'un menu « Ouvrir un fichier récent », vous pourriez modifier les options de menu en fonction de la liste actuelle de documents affichés récemment.

Si vous supprimez l'option de menu dont le raccourci clavier a déclenché un événement `preparing`, l'interaction de menu est annulée et aucun événement `select` n'est distribué.

Les propriétés `target` et `currentTarget` de l'événement correspondent toutes deux à l'objet auquel est associé l'écouteur, c'est-à-dire au menu en tant que tel ou à l'une de ses options.

L'événement `preparing` est distribué avant l'événement `displaying`. Vous écoutez généralement l'un des deux événements, mais non les deux à la fois.

Sélection : lorsque l'utilisateur choisit une commande, l'élément distribue un événement `select` à tout écouteur enregistré. Les sous-menus ou les séparateurs ne distribuent jamais d'événements `select` car il est impossible de les sélectionner.

Un événement `select` remonte d'un élément de menu vers son menu conteneur ou jusqu'au menu racine. Vous pouvez écouter les événements `select` directement sur un élément ou plus haut dans la structure de menu. Lorsque vous écoutez un événement `select` sur un menu, vous pouvez identifier l'élément sélectionné à l'aide de la propriété `target` de l'événement. Lorsque l'événement remonte dans la hiérarchie de menu, la propriété `currentTarget` de l'objet événement identifie l'objet menu actuel.

Remarque : *les objets ContextMenu et ContextMenuItem distribuent des événements* `menuItemSelect` *et* `menuSelect`*, ainsi que des événements* `select`*,* `preparing` *et* `displaying`*.*

Equivalents clavier des commandes de menus natifs (AIR)

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez affecter un équivalent clavier (parfois appelé accélérateur) à une commande de menu. L'élément de menu distribue un événement `select` à tout écouteur enregistré lorsque l'utilisateur appuie sur la touche ou la combinaison de touches. Le menu contenant l'élément doit appartenir au menu de l'application ou de la fenêtre active pour la commande à appeler.

L'équivalent clavier se compose de deux parties : une chaîne représentant la touche principale et un tableau de touches de modification sur lesquelles il est également nécessaire d'appuyer. Pour affecter une touche principale, définissez la propriété `keyEquivalent` de l'élément de menu sur la chaîne à caractère unique correspondant à cette touche. Si vous utilisez une majuscule, la touche Maj est automatiquement ajoutée au tableau de touches de modification.

Sous Mac OS X, la touche Commande (`Keyboard.COMMAND`) est la touche de modification par défaut. Sous Windows et Linux, il s'agit de la touche Ctrl (`Keyboard.CONTROL`). Ces touches par défaut sont automatiquement ajoutées au tableau de touches de modification. Pour utiliser d'autres touches de modification, affectez un nouveau tableau contenant les codes de touche souhaités à la propriété `keyEquivalentModifiers`. Le tableau par défaut est remplacé. Que vous utilisiez les touches de modification par défaut ou votre propre tableau de touches de modification, la touche Maj est ajoutée si la chaîne que vous affectez à la propriété `keyEquivalent` est une lettre majuscule. Les constantes correspondant aux codes de touche à utiliser pour les touches de modification sont définies dans la classe Keyboard.

La chaîne d'équivalent clavier affectée est automatiquement affichée en regard du nom de l'élément de menu. Le format varie selon le système d'exploitation et les préférences système de l'utilisateur.

Remarque : *si vous affectez la valeur* `Keyboard.COMMAND` *à un tableau de touches de modification sous Windows, aucun équivalent clavier n'est affiché dans le menu. En revanche, il est impératif d'utiliser la touche Ctrl pour activer la commande de menu.*

L'exemple suivant affecte l'équivalent clavier `Ctrl+Maj+G` à un élément de menu :

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

L'exemple suivant affecte l'équivalent clavier `Ctrl+Maj+G` en définissant directement le tableau de modification :

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

Remarque : *les équivalents clavier sont uniquement déclenchés pour les menus d'application et de fenêtre. Un équivalent clavier ajouté à un menu contextuel ou à un menu en incrustation est indiqué dans le libellé du menu, mais la commande de menu associée n'est jamais appelée.*

Mnémoniques (AIR)

Adobe AIR 1.0 et les versions ultérieures

Les mnémoniques font partie de l'interface clavier du système d'exploitation pour les menus. Linux, Mac OS X et Windows permettent d'ouvrir des menus et de sélectionner des commandes par le biais du clavier mais il existe quelques petites différences.

Sous Mac OS X, l'utilisateur saisit la ou les deux premières lettres du menu ou de la commande et appuie sur Entrée. La propriété `mnemonicIndex` est ignorée.

Sous Windows, seule une lettre est significative. Par défaut, elle correspond au premier caractère du libellé mais si vous affectez une mnémonique à un élément de menu, la lettre désignée devient le caractère significatif. Si deux éléments d'un menu possèdent le même caractère significatif (qu'une mnémonique ait été affectée ou non), l'interaction avec le menu à partir du clavier change légèrement. Plutôt que d'appuyer sur une seule lettre pour sélectionner le menu ou la commande, l'utilisateur appuie sur la lettre autant de fois que nécessaire pour mettre en évidence l'élément souhaité, puis il appuie sur la touche Entrée pour confirmer la sélection. Pour garantir un comportement cohérent, il est préférable d'affecter une seule mnémonique à chaque élément d'un menu sous Windows.

Sous Linux, aucune mnémonique n'est fournie par défaut. Pour fournir une mnémonique, vous devez définir la valeur de la propriété `mnemonicIndex` d'un élément de menu.

Spécifiez le caractère mnémonique en tant qu'index dans la chaîne de libellé. L'index du premier caractère d'un libellé est 0. Ainsi pour utiliser le « r » comme mnémonique d'un menu intitulé « Format », vous définissez la propriété `mnemonicIndex` sur 2.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

Etat d'un élément de menu

Adobe AIR 1.0 et les versions ultérieures

Les éléments de menu possèdent deux propriétés d'état : `checked` (coché) et `enabled` (activé) :

checked Définissez cette propriété sur `true` pour afficher une coche en regard du libellé de l'élément.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

enabled Faites basculer la valeur entre `true` et `false` pour vérifier si la commande est activée. Les éléments désactivés sont affichés « en grisé » et ne distribuent pas d'événements `select`.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

Rattachement d'un objet à un élément de menu

Adobe AIR 1.0 et les versions ultérieures

La propriété `data` de la classe NativeMenuItem vous permet de référencer un objet arbitraire dans chaque élément. Par exemple, dans un menu « Ouvrir un fichier récent », vous pourriez affecter l'objet File associé à chaque document à chaque élément de menu.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

Création de menus natifs (AIR)

Adobe AIR 1.0 et les versions ultérieures

Cette rubrique explique comment créer les différents types de menus natifs pris en charge par AIR.

Création d'un objet menu racine

Adobe AIR 1.0 et les versions ultérieures

Pour créer un objet NativeMenu en tant que racine du menu, utilisez le constructeur NativeMenu :

```
var root:NativeMenu = new NativeMenu();
```

Dans les menus d'application et de fenêtre, le menu racine représente la barre de menus et doit uniquement contenir des éléments donnant accès à des sous-menus. Les menus en incrustation et les menus contextuels ne possèdent pas de barre de menus. Le menu racine peut donc contenir des commandes et des séparateurs, ainsi que des sous-menus.

Une fois le menu créé, vous pouvez lui ajouter des éléments de menu. Les éléments figurent dans le menu dans l'ordre dans lequel vous les ajoutez, à moins que vous ne les ajoutiez à un index spécifique à l'aide de la méthode `addItemAt()` d'un objet menu.

Affectez le menu en tant que menu d'application, de fenêtre, d'icône ou contextuel ou affichez-le en tant que menu en incrustation, comme indiqué dans les sections suivantes :

Définition d'un menu d'application ou de fenêtre

Le code doit prendre en charge à la fois les menus d'application (gérés sous Mac OS) et les menus de fenêtre (gérés sous d'autres systèmes d'exploitation).

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

Remarque : *Mac OS définit, pour chaque application, un menu contenant des éléments standard. L'affectation d'un nouvel objet NativeMenu à la propriété* `menu` *de l'objet NativeApplication remplace le menu standard. Libre à vous de conserver ce menu standard plutôt que de le remplacer.*

Adobe Flex contient une classe FlexNativeMenu qui permet de créer facilement des menus pris en charge sur toutes les plateformes. Si vous utilisez Flex Framework, faites appel aux classes FlexNativeMenu au lieu de la classe NativeMenu.

Définition d'un menu contextuel associé à un objet interactif

```
interactiveObject.contextMenu = root;
```

Définition d'un menu d'icône du Dock ou d'un menu d'icône de la barre d'état système

Le code doit prendre en charge à la fois les menus d'application (gérés sous Mac OS) et les menus de fenêtre (gérés sous d'autres systèmes d'exploitation).

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

Remarque : *Mac OS X définit un menu standard pour l'icône du Dock d'une application. Lorsque vous affectez un nouvel objet NativeMenu à la propriété menu de l'objet DockIcon, les éléments de ce menu sont affichés au-dessus des éléments standard. Il est impossible de supprimer ou de modifier les éléments standard ou d'y accéder.*

Affichage d'un menu en incrustation

```
root.display(stage, x, y);
```

Voir aussi

Développement d'applications AIR multiplateformes

Création d'un sous-menu

Adobe AIR 1.0 et les versions ultérieures

Pour créer un sous-menu, vous ajoutez un objet NativeMenuItem au menu parent, puis vous affectez l'objet NativeMenu définissant le sous-menu sur la propriété `submenu` de l'élément. AIR vous propose deux méthodes de création des éléments de sous-menu et de l'objet menu associé :

Vous pouvez créer un élément de menu et son objet menu associé en une seule opération à l'aide de la méthode `addSubmenu()` :

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

Vous pouvez aussi créer l'élément de menu et ensuite seulement affecter l'objet menu à sa propriété `submenu` :

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

Création d'une commande de menu

Adobe AIR 1.0 et les versions ultérieures

Pour créer une commande de menu, ajoutez un objet NativeMenuItem à un menu et ajoutez un écouteur d'événement référençant la fonction qui met en œuvre la commande de menu :

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

Vous pouvez écouter l'événement `select` sur l'élément commande même (comme illustré dans l'exemple) ou sur un objet menu parent.

Remarque : *les éléments de menu qui représentent des sous-menus et des séparateurs ne distribuent pas d'événements* `select` *et vous ne pouvez donc pas les utiliser en tant que commandes.*

Création d'un séparateur de menu

Adobe AIR 1.0 et les versions ultérieures

Pour créer un séparateur, créez un objet NativeMenuItem, en définissant le paramètre `isSeparator` sur `true` dans le constructeur. Ajoutez ensuite le séparateur au menu à l'emplacement approprié :

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

Le libellé spécifié pour le séparateur, le cas échéant, n'est pas affiché.

A propos des menus contextuels dans un contenu HTML (AIR)

Adobe AIR 1.0 et les versions ultérieures

Dans un contenu HTML affiché à l'aide de l'objet HTMLLoader, l'événement `contextmenu` permet d'afficher un menu contextuel. Par défaut, un menu contextuel est affiché automatiquement lorsque l'utilisateur appelle l'événement contextmenu sur le texte sélectionné (en cliquant avec le bouton droit de la souris ou en cliquant tout en appuyant sur la touche Commande). Pour empêcher l'affichage du menu par défaut, écoutez l'événement `contextmenu` et appelez la méthode `preventDefault()` de l'objet événement :

```
function showContextMenu(event){
    event.preventDefault();
}
```

Vous pouvez ensuite ouvrir un menu contextuel personnalisé à l'aide des techniques DHTML ou en affichant un menu contextuel natif AIR. Dans l'exemple suivant, un menu contextuel natif s'affiche sur appel de la méthode `display()` du menu en réponse à l'événement HTML `contextmenu` :

```
 <html>
<head>
<script src="AIRAliases.js" language="JavaScript" type="text/javascript"></script>
<script language="javascript" type="text/javascript">

function showContextMenu(event){
    event.preventDefault();
    contextMenu.display(window.nativeWindow.stage, event.clientX, event.clientY);
}

function createContextMenu(){
    var menu = new air.NativeMenu();
    var command = menu.addItem(new air.NativeMenuItem("Custom command"));
    command.addEventListener(air.Event.SELECT, onCommand);
    return menu;
}

function onCommand(){
    air.trace("Context command invoked.");
}

var contextMenu = createContextMenu();
</script>
</head>
<body>
<p oncontextmenu="showContextMenu(event)" style="-khtml-user-select:auto;">Custom context
menu.</p>
</body>
</html>
```

Affichage de menus natifs en incrustation (AIR)

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez afficher tout objet NativeMenu à tout moment et à tout emplacement au-dessus d'une fenêtre en appelant la méthode `display()` du menu. Cette méthode exigeant une référence à la scène, seul le contenu figurant dans le sandbox de l'application peut afficher un menu en incrustation.

La méthode suivante affiche le menu défini par l'objet NativeMenu `popupMenu` en réponse à un clic de souris :

```
 private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

Remarque : *il est inutile d'afficher le menu directement en réponse à un événement. N'importe quelle méthode peut appeler la fonction* `display()`.

Gestion des événements de menu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un menu distribue des événements lorsque l'utilisateur sélectionne le menu ou l'un de ses éléments.

Récapitulatif des événements associés aux classes de menu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Ajoutez des écouteurs d'événement à des menus ou à des éléments individuels pour gérer les événements de menu.

Object	Evénements distribués
NativeMenu (AIR)	Event.PREPARING (Adobe AIR 2.6 et les versions ultérieures) Event.DISPLAYING Event.SELECT (propagé à partir des sous-menus et éléments enfant)
NativeMenuItem (AIR)	Event.PREPARING (Adobe AIR 2.6 et les versions ultérieures) Event.SELECT Event.DISPLAYING (propagé à partir du menu parent)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT (AIR)

Evénements de menu select

Adobe AIR 1.0 et les versions ultérieures

Pour gérer un clic sur un élément de menu, ajoutez à l'objet NativeMenuItem un écouteur d'événement relatif à l'événement `select` :

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

Comme les événements `select` remontent jusqu'aux menus conteneur, vous pouvez aussi les écouter sur un menu parent. Lorsque vous écoutez au niveau d'un menu, vous pouvez utiliser la propriété `target` de l'objet événement pour déterminer la commande de menu sélectionnée. L'exemple suivant suit la trace du libellé de la commande sélectionnée :

```
 var colorMenuItem:NativeMenuItem = new NativeMenuItem("Choose a color");
var colorMenu:NativeMenu = new NativeMenu();
colorMenuItem.submenu = colorMenu;

var red:NativeMenuItem = new NativeMenuItem("Red");
var green:NativeMenuItem = new NativeMenuItem("Green");
var blue:NativeMenuItem = new NativeMenuItem("Blue");
colorMenu.addItem(red);
colorMenu.addItem(green);
colorMenu.addItem(blue);

if(NativeApplication.supportsMenu){
    NativeApplication.nativeApplication.menu.addItem(colorMenuItem);
    NativeApplication.nativeApplication.menu.addEventListener(Event.SELECT, colorChoice);
} else if (NativeWindow.supportsMenu){
    var windowMenu:NativeMenu = new NativeMenu();
    this.stage.nativeWindow.menu = windowMenu;
    windowMenu.addItem(colorMenuItem);
    windowMenu.addEventListener(Event.SELECT, colorChoice);
}

function colorChoice(event:Event):void {
    var menuItem:NativeMenuItem = event.target as NativeMenuItem;
    trace(menuItem.label + " has been selected");
}
```

Si vous utilisez la classe ContextMenuItem, vous pouvez écouter l'événement `select` ou `menuItemSelect`. L'événement `menuItemSelect` donne des informations complémentaires sur l'objet auquel appartient le menu contextuel, mais ne remonte pas jusqu'aux menus conteneur.

Evénements de menu displaying

Adobe AIR 1.0 et les versions ultérieures

Pour gérer l'ouverture d'un menu, vous pouvez ajouter un événement `displaying`, qui est distribué avant l'affichage d'un menu. L'événement displaying permet de mettre à jour le menu, par exemple en ajoutant ou supprimant des éléments, ou en actualisant l'état, activé ou coché, d'éléments individuels. Vous pouvez également écouter l'événement `menuSelect` à partir d'un objet ContextMenu.

Dans AIR 2.6 et les versions ultérieures, vous disposez de l'événement `preparing` pour mettre à jour un menu en réponse à l'affichage d'un menu ou à la sélection d'une option par le biais d'un raccourci clavier.

Exemple de menu natif : menu de fenêtre et d'application (AIR)

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée le menu affiché à la section « Structure de menu natif (AIR) » à la page 657.

De par sa conception, ce menu fonctionne sous Windows, qui prend uniquement en charge les menus de fenêtre, et sous Mac OS X, qui prend uniquement en charge les menus d'application. Pour faire la distinction, le constructeur de la classe MenuExample vérifie les propriétés `supportsMenu` statiques des classes NativeWindow et NativeApplication. Si `NativeWindow.supportsMenu` est défini sur `true`, le constructeur crée un objet NativeMenu pour la fenêtre, puis crée et ajoute les sous-menus File et Edit. Si `NativeApplication.supportsMenu` est défini sur `true`, le constructeur crée les menus File et Edit, et les ajoute au menu existant que propose le système d'exploitation Mac OS X.

L'exemple suivant illustre également la gestion des événements de menu. L'événement `select` est géré au niveau des éléments et au niveau du menu. Chaque menu du chaînage allant du menu contenant l'élément sélectionné au menu racine répond à l'événement `select`. L'événement `displaying` est utilisé avec le menu « Ouvrir un fichier récent ». Juste avant l'ouverture du menu, ses éléments sont actualisés à partir du tableau recentDocuments (qui, dans cet exemple, reste inchangé). Bien que cette procédure ne soit pas illustrée dans cet exemple, vous pouvez également écouter des événements `displaying` sur des éléments individuels.

```
package {
    import flash.display.NativeMenu;
    import flash.display.NativeMenuItem;
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.filesystem.File;
    import flash.desktop.NativeApplication;

    public class MenuExample extends Sprite
    {
        private var recentDocuments:Array =
            new Array(new File("app-storage:/GreatGatsby.pdf"),
                      new File("app-storage:/WarAndPeace.pdf"),
                      new File("app-storage:/Iliad.pdf"));

        public function MenuExample()
        {
            var fileMenu:NativeMenuItem;
            var editMenu:NativeMenuItem;

            if (NativeWindow.supportsMenu){
                stage.nativeWindow.menu = new NativeMenu();
                stage.nativeWindow.menu.addEventListener(Event.SELECT, selectCommandMenu);
                fileMenu = stage.nativeWindow.menu.addItem(new NativeMenuItem("File"));
                fileMenu.submenu = createFileMenu();
                editMenu = stage.nativeWindow.menu.addItem(new NativeMenuItem("Edit"));
                editMenu.submenu = createEditMenu();
            }

            if (NativeApplication.supportsMenu){
                NativeApplication.nativeApplication.menu.addEventListener(Event.SELECT,
selectCommandMenu);
                fileMenu = NativeApplication.nativeApplication.menu.addItem(new
NativeMenuItem("File"));
                fileMenu.submenu = createFileMenu();
                editMenu = NativeApplication.nativeApplication.menu.addItem(new
NativeMenuItem("Edit"));
                editMenu.submenu = createEditMenu();
            }
        }
```

```
public function createFileMenu():NativeMenu {
    var fileMenu:NativeMenu = new NativeMenu();
    fileMenu.addEventListener(Event.SELECT, selectCommandMenu);

    var newCommand:NativeMenuItem = fileMenu.addItem(new NativeMenuItem("New"));
    newCommand.addEventListener(Event.SELECT, selectCommand);
    var saveCommand:NativeMenuItem = fileMenu.addItem(new NativeMenuItem("Save"));
    saveCommand.addEventListener(Event.SELECT, selectCommand);
    var openRecentMenu:NativeMenuItem =
            fileMenu.addItem(new NativeMenuItem("Open Recent"));
    openRecentMenu.submenu = new NativeMenu();
    openRecentMenu.submenu.addEventListener(Event.DISPLAYING,
                                    updateRecentDocumentMenu);
    openRecentMenu.submenu.addEventListener(Event.SELECT, selectCommandMenu);

    return fileMenu;
}

public function createEditMenu():NativeMenu {
    var editMenu:NativeMenu = new NativeMenu();
    editMenu.addEventListener(Event.SELECT, selectCommandMenu);

    var copyCommand:NativeMenuItem = editMenu.addItem(new NativeMenuItem("Copy"));
    copyCommand.addEventListener(Event.SELECT, selectCommand);
    copyCommand.keyEquivalent = "c";
    var pasteCommand:NativeMenuItem =
            editMenu.addItem(new NativeMenuItem("Paste"));
    pasteCommand.addEventListener(Event.SELECT, selectCommand);
    pasteCommand.keyEquivalent = "v";
    editMenu.addItem(new NativeMenuItem("", true));
    var preferencesCommand:NativeMenuItem =
            editMenu.addItem(new NativeMenuItem("Preferences"));
    preferencesCommand.addEventListener(Event.SELECT, selectCommand);

    return editMenu;
}

private function updateRecentDocumentMenu(event:Event):void {
    trace("Updating recent document menu.");
    var docMenu:NativeMenu = NativeMenu(event.target);

    for each (var item:NativeMenuItem in docMenu.items) {
        docMenu.removeItem(item);
    }

    for each (var file:File in recentDocuments) {
        var menuItem:NativeMenuItem =
                docMenu.addItem(new NativeMenuItem(file.name));
        menuItem.data = file;
        menuItem.addEventListener(Event.SELECT, selectRecentDocument);
    }
}

private function selectRecentDocument(event:Event):void {
    trace("Selected recent document: " + event.target.data.name);
}
```

```
        private function selectCommand(event:Event):void {
            trace("Selected command: " + event.target.label);
        }

        private function selectCommandMenu(event:Event):void {
            if (event.currentTarget.parent != null) {
                var menuItem:NativeMenuItem =
                        findItemForMenu(NativeMenu(event.currentTarget));
                if (menuItem != null) {
                    trace("Select event for \"" +
                            event.target.label +
                            "\" command handled by menu: " +
                            menuItem.label);
                }
            } else {
                trace("Select event for \"" +
                        event.target.label +
                        "\" command handled by root menu.");
            }
        }

        private function findItemForMenu(menu:NativeMenu):NativeMenuItem {
            for each (var item:NativeMenuItem in menu.parent.items) {
                if (item != null) {
                    if (item.submenu == menu) {
                        return item;
                    }
                }
            }
            return null;
        }
    }
}
```

Chapitre 37 : Icônes de la barre des tâches dans AIR

Adobe AIR 1.0 et les versions ultérieures

La plupart des systèmes d'exploitation disposent d'une barre des tâches, comme le Dock de Mac OS X, pouvant contenir une icône en vue de représenter une application. Adobe® AIR® dispose d'une interface permettant d'interagir avec l'icône de la barre des tâches de l'application par le biais de la propriété `NativeApplication.nativeApplication.icon`.

- Utilisation des icônes de la barre d'état système et du Dock (Flex)
- Utilisation des icônes de la barre d'état système et du Dock (Flash)

Voir aussi

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

A propos des icônes de la barre des tâches

Adobe AIR 1.0 et les versions ultérieures

AIR crée l'objet `NativeApplication.nativeApplication.icon` automatiquement. Selon le système d'exploitation, le type d'objet est soit DockIcon soit SystemTrayIcon. Les propriétés `NativeApplication.supportsDockIcon` et `NativeApplication.supportsSystemTrayIcon` vous permettent de déterminer les sous-classes InteractiveIcon prises en charge par AIR sur le système d'exploitation actuel. La classe de base InteractiveIcon fournit les propriétés `width`, `height` et `bitmaps`, qui vous permettent de modifier l'image utilisée pour l'icône. En revanche, l'accès aux propriétés spécifiques à l'objet DockIcon ou SystemTrayIcon sur le mauvais système d'exploitation génère une erreur d'exécution.

Pour définir ou modifier l'image utilisée pour une icône, créez un tableau contenant une ou plusieurs images, puis affectez-le à la propriété `NativeApplication.nativeApplication.icon.bitmaps`. La taille des icônes de la barre des tâches peut varier d'un système d'exploitation à l'autre. Pour éviter la dégradation de l'image provoquée par la mise à l'échelle, vous pouvez ajouter plusieurs tailles d'image au tableau `bitmaps`. Si vous fournissez plusieurs images, AIR sélectionne la taille la plus proche de la taille d'affichage actuelle de l'icône de la barre des tâches et, le cas échéant, met ces images à l'échelle. Dans l'exemple suivant, l'image d'une icône de la barre des tâches est définie à l'aide de deux images :

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

Pour modifier l'image d'une icône, affectez un tableau contenant la ou les nouvelles images à la propriété `bitmaps`. Vous pouvez animer l'icône en modifiant l'image en réponse à un événement `enterFrame` ou `timer`.

Pour supprimer l'icône de la zone de notification sous Windows et Linux, ou pour restaurer l'apparence de l'icône par défaut sous Mac OS X, définissez `bitmaps` sur un tableau vide :

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

Icônes du Dock

Adobe AIR 1.0 et les versions ultérieures

AIR prend en charge les icônes du Dock lorsque la propriété `NativeApplication.supportsDockIcon` est définie sur `true`. La propriété `NativeApplication.nativeApplication.icon` représente l'icône de l'application sur le Dock (et non l'icône de la fenêtre).

Remarque : *AIR ne permet pas de modifier les icônes de fenêtre sur le Dock sous Mac OS X. Par ailleurs, les modifications apportées à l'icône du Dock de l'application ne sont appliquées que lorsqu'une application est en cours d'exécution ; l'icône retrouve son aspect normal lorsque vous quittez l'application.*

Menus d'icônes du Dock

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez ajouter des commandes au menu du Dock standard en créant un objet NativeMenu contenant les commandes, puis en l'affectant à la propriété `NativeApplication.nativeApplication.icon.menu`. Les éléments du menu s'affichent au-dessus des options de menu de l'icône du Dock standard.

Rebond de l'icône du Dock

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez faire rebondir l'icône du Dock en appelant la méthode `NativeApplication.nativeApplication.icon.bounce()`. Si vous définissez le paramètre `bounce() priority` sur informational, l'icône rebondit une fois. Si vous définissez ce paramètre sur critical, l'icône rebondit jusqu'à ce que l'utilisateur active l'application. Les constantes du paramètre `priority` sont définies dans la classe NotificationType.

Remarque : *l'icône ne rebondit pas si l'application est déjà active.*

Evénements de l'icône du Dock

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous cliquez sur l'icône du Dock, l'objet NativeApplication distribue un événement `invoke`. Si l'application n'est pas en cours d'exécution, le système lance l'application. Sinon, l'événement `invoke` est renvoyé à l'occurrence de l'application en cours d'exécution.

Icônes de la barre d'état système

Adobe AIR 1.0 et les versions ultérieures

AIR prend en charge les icônes de la barre d'état système lorsque la propriété `NativeApplication.supportsSystemTrayIcon` est définie sur `true`, ce qui n'est actuellement le cas que sous Windows et la plupart des distributions Linux. Sous Windows et Linux, les icônes de la barre d'état système s'affichent dans la zone de notification de la barre des tâches. Par défaut, aucune icône n'est affichée. Pour afficher une icône, affectez un tableau contenant des objets BitmapData à la propriété `bitmaps` de l'icône. Pour modifier l'image d'une icône, affectez un tableau contenant les nouvelles images à la propriété `bitmaps`. Pour supprimer l'icône, définissez la propriété `bitmaps` sur `null`.

Menus de l'icône de la barre d'état système

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez ajouter un menu à l'icône de la barre d'état système en créant un objet NativeMenu, puis en l'affectant à la propriété `NativeApplication.nativeApplication.icon.menu` (le système d'exploitation ne fournit aucun menu par défaut). Pour accéder au menu de l'icône de la barre d'état système, cliquez sur l'icône avec le bouton droit de la souris.

Info-bulles de l'icône de la barre d'état système

Adobe AIR 1.0 et les versions ultérieures

Pour ajouter une info-bulle à une icône, définissez la propriété tooltip :

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

Evénements de l'icône de la barre d'état système

Adobe AIR 1.0 et les versions ultérieures

L'objet SystemTrayIcon référencé par la propriété NativeApplication.nativeApplication.icon distribue un événement ScreenMouseEvent pour les événements `click`, `mouseDown`, `mouseUp`, `rightClick`, `rightMouseDown` et `rightMouseUp`. Vous pouvez utiliser ces événements, ainsi que le menu d'une icône, pour autoriser les utilisateurs à interagir avec votre application lorsque celle-ci ne dispose pas de fenêtres visibles.

Exemple : Création d'une application ne disposant d'aucune fenêtre

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant crée une application AIR qui dispose d'une icône de la barre d'état système, mais d'aucune fenêtre visible. (Ne définissez pas la propriété `visible` de l'application sur `true` dans le descripteur d'application, sous peine que la fenêtre ne soit visible au démarrage de l'application.)

```
package
{
    import flash.display.Loader;
    import flash.display.NativeMenu;
    import flash.display.NativeMenuItem;
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.desktop.DockIcon;
    import flash.desktop.SystemTrayIcon;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.desktop.NativeApplication;

    public class SysTrayApp extends Sprite
    {
        public function SysTrayApp():void{
            NativeApplication.nativeApplication.autoExit = false;
            var icon:Loader = new Loader();
            var iconMenu:NativeMenu = new NativeMenu();
            var exitCommand:NativeMenuItem = iconMenu.addItem(new NativeMenuItem("Exit"));
                exitCommand.addEventListener(Event.SELECT, function(event:Event):void {
                    NativeApplication.nativeApplication.icon.bitmaps = [];
                    NativeApplication.nativeApplication.exit();
                });

            if (NativeApplication.supportsSystemTrayIcon) {
                NativeApplication.nativeApplication.autoExit = false;
                icon.contentLoaderInfo.addEventListener(Event.COMPLETE, iconLoadComplete);
                icon.load(new URLRequest("icons/AIRApp_16.png"));

                var systray:SystemTrayIcon =
                    NativeApplication.nativeApplication.icon as SystemTrayIcon;
                systray.tooltip = "AIR application";
                systray.menu = iconMenu;
            }

            if (NativeApplication.supportsDockIcon){
                icon.contentLoaderInfo.addEventListener(Event.COMPLETE,iconLoadComplete);
                icon.load(new URLRequest("icons/AIRApp_128.png"));
                var dock:DockIcon = NativeApplication.nativeApplication.icon as DockIcon;
                dock.menu = iconMenu;
            }
        }

        private function iconLoadComplete(event:Event):void
        {
            NativeApplication.nativeApplication.icon.bitmaps =
                [event.target.content.bitmapData];
        }
    }
}
```

Remarque : *si vous utilisez le composant WindowedApplication de Flex, vous devez définir l'attribut* `visible` *de la balise WindowedApplication sur* `false`*. Cet attribut prime sur le paramètre défini dans le descripteur d'application.*

Remarque : *dans cet exemple, nous supposons qu'il existe des fichiers image nommés* `AIRApp_16.png` *et* `AIRApp_128.png` *dans un sous-répertoire* `icons` *de l'application. (Les fichiers d'icône d'exemple, que vous pouvez copier dans le dossier de votre projet, sont inclus dans le kit de développement AIR.)*

Icônes et boutons de la barre des tâches de la fenêtre

Adobe AIR 1.0 et les versions ultérieures

Les icônes des fenêtres s'affichent normalement dans une zone de la fenêtre, appelée barre des tâches ou Dock, pour permettre aux utilisateurs d'accéder aisément à l'arrière-plan ou aux fenêtre minimisées. Le Dock de Mac OS X affiche l'icône correspondant à votre application, ainsi qu'une icône pour chaque fenêtre minimisée. Les barres des tâches de Microsoft Windows et Linux affichent un bouton contenant l'icône et le titre du programme de chaque fenêtre standard dans votre application.

Mise en surbrillance du bouton de la fenêtre dans la barre des tâches

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'une fenêtre se trouve dans le chrome, vous pouvez informer l'utilisateur qu'un événement important ayant trait à la fenêtre s'est produit. Sous Mac OS X, vous pouvez informer l'utilisateur en faisant rebondir l'icône de l'application du Dock (comme décrit à la section « Rebond de l'icône du Dock » à la page 671). Sous Windows et Linux, vous pouvez mettre en surbrillance le bouton de la barre des tâches de la fenêtre en appelant la méthode `notifyUser()` de l'occurrence de NativeWindow. Le paramètre `type` transmis à la méthode détermine l'urgence de la notification :

- `NotificationType.CRITICAL` : l'icône de la fenêtre clignote jusqu'à ce que l'utilisateur ramène la fenêtre au premier plan.
- `NotificationType.INFORMATIONAL` : l'icône de la fenêtre est mise en surbrillance et change de couleur.

  Remarque : *sous Linux, seul le type de notification informationnel est pris en charge. La transmission de la valeur du type à la fonction* `notifyUser()` *a le même effet.*

  L'instruction suivante met en surbrillance le bouton de la barre des tâches d'une fenêtre :

  ```
  stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
  ```

  L'appel de la méthode `NativeWindow.notifyUser()` d'un système d'exploitation qui ne prend pas en charge la notification au niveau de la fenêtre n'a aucun effet. Utilisez la propriété `NativeWindow.supportsNotification` pour déterminer si la notification de fenêtres est prise en charge.

Création de fenêtres sans icônes ni boutons dans la barre des tâches

Adobe AIR 1.0 et les versions ultérieures

Sous Windows, les fenêtres de type *utilitaire* ou *légère* ne s'affichent pas dans la barre des tâches. Les fenêtres invisibles n'apparaissent pas non plus dans la barre des tâches.

Etant donné que la fenêtre initiale est nécessairement de type *normale*, pour créer une application dont aucune fenêtre ne s'affiche dans la barre des tâches, vous devez fermer la fenêtre initiale ou la laisser invisible. Pour fermer toutes les fenêtres de votre application sans quitter l'application, définissez la propriété `autoExit` de l'objet NativeApplication sur `false` avant de fermer la dernière fenêtre. Pour simplement éviter que la fenêtre initiale soit visible, ajoutez `<visible>false</visible>` à l'élément `<initalWindow>` du fichier descripteur d'application (et ne définissez pas la propriété `visible` sur `true`, ni n'appelez la méthode `activate()` de la fenêtre).

Dans les nouvelles fenêtres ouvertes par l'application, définissez la propriété `type` de l'objet NativeWindowInitOption transmis au constructeur de fenêtres sur `NativeWindowType.UTILITY` ou sur `NativeWindowType.LIGHTWEIGHT`.

Sous Mac OS X, les fenêtres minimisées apparaissent sur la barre des tâches du Dock. Vous pouvez désactiver l'affichage de l'icône minimisée en masquant la fenêtre au lieu de la minimiser. L'exemple suivant écoute un événement de changement `nativeWindowDisplayState` et l'annule si la fenêtre est minimisée. Le gestionnaire définit la propriété `visible` de la fenêtre sur `false` :

```
private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

Sous Mac OS X, si une fenêtre est minimisée sur le Dock lorsque vous définissez la propriété `visible` sur `false`, l'icône du Dock n'est pas supprimée. L'utilisateur peut toujours cliquer sur l'icône pour faire réapparaître la fenêtre.

Chapitre 38 : Utilisation du système de fichiers

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash® Player propose des fonctionnalités de lecture et d'écriture de fichier de base, par le biais de la classe FileReference. Pour des raisons de sécurité, l'utilisateur doit systématiquement accorder son autorisation pour que vous puissiez lire ou écrire un fichier dans Flash Player.

Adobe® AIR® assure un accès plus complet au système de fichiers de l'ordinateur hôte que Flash Player. L'API du système de fichiers AIR vous permet d'accéder aux fichiers et aux répertoires et de les gérer, de créer des fichiers et des répertoires, d'écrire des données dans les fichiers, etc.

Voir aussi

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

Utilisation de la classe FileReference

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet FileReference représente un fichier de données stocké sur un client ou un serveur. Les méthodes de la classe FileReference permettent à votre application de charger et d'enregistrer localement des fichiers de données et de transférer ces derniers entre la machine locale et un serveur distant.

La classe FileReference propose deux approches distinctes pour charger, transférer et enregistrer les fichiers de données. Depuis son introduction, la classe FileReference comprend les méthodes `browse()`, `upload()` et `download()`. La méthode `browse()` permet à l'utilisateur de sélectionner un fichier. La méthode `upload()` permet de transférer les données du fichier vers un serveur distant. La méthode `download()` permet d'extraire les données du serveur et de les enregistrer dans un fichier local. Depuis Flash Player 10 et Adobe AIR 1.5, la classe FileReference intègre les méthodes `load()` et `save()`. Grâce aux méthodes `load()` et `save()`, vous pouvez également accéder aux fichiers locaux et les enregistrer directement. L'utilisation de ces méthodes est similaire aux méthodes du même nom dont disposent les classes URLLoader et Loader.

Remarque : *la classe File, qui étend la classe FileReference, et la classe FileStream proposent d'autres fonctions de manipulation des fichiers et du système de fichiers local. Les classes File et FileStream sont prises en charge dans AIR uniquement et non dans Flash Player.*

FileReference, classe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Chaque objet FileReference représente un fichier de données local hébergé sur la machine locale. Les propriétés de la classe FileReference contiennent des informations sur la taille, le type, le nom, l'extension, le créateur, la date de création et la date de modification du fichier.

Remarque : *la propriété* `creator` *est prise en charge sous Mac OS uniquement. Toutes les autres plates-formes renvoient la valeur* `null`*.*

Remarque : *la propriété* `extension` *est prise en charge par Adobe AIR uniquement.*

Pour créer une occurrence de la classe FileReference, procédez comme suit, au choix :

- Utilisez l'opérateur `new`, comme indiqué dans le code suivant :

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- Appelez la méthode `FileReferenceList.browse()`, qui ouvre une boîte de dialogue et invite l'utilisateur à sélectionner un ou plusieurs fichiers à télécharger. Elle crée ensuite un tableau d'objets FileReference si l'utilisateur réussit à sélectionner un ou plusieurs fichiers.

Une fois l'objet FileReference créé, vous pouvez procéder comme suit :

- Appelez la méthode `FileReference.browse()`, qui ouvre une boîte de dialogue et invite l'utilisateur à sélectionner un fichier unique dans le système de fichiers local. Cette opération est généralement effectuée avant d'appeler la méthode `FileReference.upload()` ou `FileReference.load()`. La méthode `FileReference.upload()` charge le fichier sur un serveur distant. La méthode `FileReference.load()` ouvre un fichier local.
- Appelez la méthode `FileReference.download()`. La méthode `download` ouvre une boîte de dialogue qui permet à l'utilisateur de sélectionner l'emplacement d'enregistrement d'un nouveau fichier. Elle télécharge ensuite les données du serveur et les stocke dans le nouveau fichier.
- Appelez la méthode `FileReference.load()`. Cette méthode commence le chargement de données à partir d'un fichier précédemment sélectionné à l'aide de la méthode `browse()`. Il est impossible d'appeler la méthode `load()` tant que l'opération `browse()` n'est pas terminée (c'est-à-dire lorsque l'utilisateur sélectionne un fichier).
- Appelez la méthode `FileReference.save()`. Cette méthode ouvre une boîte de dialogue et invite l'utilisateur à sélectionner un emplacement de fichier unique sur le système de fichiers local. Elle permet ensuite d'enregistrer les données à l'emplacement spécifié.

Remarque : *vous ne pouvez exécuter qu'une seule méthode* `browse()`*,* `download()` *ou* `save()` *à la fois, car une seule boîte de dialogue peut être ouverte à un moment donné.*

Les propriétés de l'objet FileReference, telles que `name`, `size` ou `modificationDate`, ne sont pas définies tant que l'un des événements suivants ne s'est pas produit :

- La méthode `FileReference.browse()` ou `FileReferenceList.browse()` a été appelée et l'utilisateur a sélectionné un fichier dans la boîte de dialogue.
- La méthode `FileReference.download()` a été appelée et l'utilisateur a stipulé un nouvel emplacement de fichier par le biais de la boîte de dialogue.

Remarque : *lors d'un téléchargement, seule la propriété* `FileReference.name` *est renseignée avant la fin du téléchargement. Une fois que le fichier est téléchargé, toutes les propriétés sont disponibles.*

Lors de l'exécution des appels de la méthode `FileReference.browse()`, `FileReferenceList.browse()`, `FileReference.download()`, `FileReference.load()` ou `FileReference.save()`, la plupart des lecteurs poursuivent la lecture du fichier SWF, ainsi que la distribution d'événements et l'exécution du code.

Pour les opérations de chargement ou téléchargement, un fichier SWF peut uniquement accéder aux fichiers de son propre domaine, ce qui comprend tous les domaines spécifiés par un fichier de régulation. Vous devez placer un fichier de régulation sur le serveur contenant le fichier, si ce serveur ne se trouve pas sur le même domaine que le fichier SWF ayant initié le chargement ou le téléchargement.

Voir FileReference.

Chargement de données à partir d'un fichier

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `FileReference.load()` vous permet de charger des données en mémoire à partir d'un fichier local.

Remarque : *le code doit tout d'abord appeler la méthode `FileReference.browse()` pour que l'utilisateur puisse sélectionner le fichier à charger. Cette restriction ne s'applique pas au contenu qui s'exécute dans le sandbox de sécurité de l'application Adobe AIR.*

La méthode `FileReference.load()` renvoie une valeur immédiatement après avoir été appelée, mais les données en cours de chargement ne sont pas disponibles tout de suite. L'objet FileReference distribue des événements pour appeler les méthodes d'écouteur à chaque étape du processus de chargement.

L'objet FileReference distribue les événements suivants pendant le processus de chargement.

- Evénement `open` (`Event.OPEN`) : distribué lorsque l'opération de chargement commence.
- Evénement `progress` (`ProgressEvent.PROGRESS`) : distribué régulièrement lorsque le fichier lit des octets de données.
- Evénement `complete` (`Event.COMPLETE`) : distribué en cas de réussite de l'opération de chargement.
- Evénement `ioError` (`IOErrorEvent.IO_ERROR`) : distribué si le processus de chargement échoue en raison d'une erreur d'entrée/sortie lors de l'ouverture ou de la lecture des données du fichier.

Une fois que l'objet FileReference distribue l'événement complete, il est possible d'accéder aux données chargées comme un élément ByteArray dans la propriété `data` de l'objet FileReference.

L'exemple suivant indique comment inviter l'utilisateur à sélectionner un fichier, puis à charger les données de ce dernier en mémoire :

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.FileFilter;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.utils.ByteArray;

    public class FileReferenceExample1 extends Sprite
    {
        private var fileRef:FileReference;
        public function FileReferenceExample1()
        {
            fileRef = new FileReference();
            fileRef.addEventListener(Event.SELECT, onFileSelected);
            fileRef.addEventListener(Event.CANCEL, onCancel);
            fileRef.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
            fileRef.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
                        onSecurityError);
            var textTypeFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)",
                        "*.txt;*.rtf");
            fileRef.browse([textTypeFilter]);
        }
        public function onFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.addEventListener(Event.COMPLETE, onComplete);
            fileRef.load();
        }

        public function onProgress(evt:ProgressEvent):void
        {
            trace("Loaded " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }

        public function onComplete(evt:Event):void
        {
            trace("File was successfully loaded.");
            trace(fileRef.data);
        }

        public function onCancel(evt:Event):void
        {
            trace("The browse request was canceled by the user.");
        }

        public function onIOError(evt:IOErrorEvent):void
        {
            trace("There was an IO Error.");
        }
        public function onSecurityError(evt:Event):void
        {
            trace("There was a security error.");
        }
    }
}
```

Le code d'exemple crée tout d'abord l'objet FileReference nommé `fileRef`, puis appelle sa méthode `browse()`. La méthode `browse` ouvre une boîte de dialogue et l'utilisateur est invité à sélectionner un fichier. Une fois le fichier sélectionné, le code appelle la méthode `onFileSelected()`. Cette méthode ajoute des écouteurs aux événements `progress` et `complete`, puis appelle la méthode `load()` de l'objet FileReference. Les autres méthodes de gestionnaire de cet exemple se contentent de générer des messages qui indiquent le déroulement de l'opération de chargement. Lorsque le chargement est terminé, l'application affiche le contenu du fichier chargé à l'aide de la méthode `trace()`.

Dans Adobe AIR, la classe FileStream propose d'autres fonctionnalités de lecture des données dans un fichier local (voir « Lecture et écriture de fichiers » à la page 715).

Enregistrement de données dans des fichiers locaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `FileReference.save()` vous permet d'enregistrer des données dans un fichier local. Elle commence par ouvrir une boîte de dialogue qui permet à l'utilisateur d'entrer un nouveau nom de fichier et l'emplacement d'enregistrement du fichier. Une fois le nom de fichier et l'emplacement sélectionnés, les données sont écrites dans le nouveau fichier. Lorsque le fichier est enregistré, les propriétés de l'objet FileReference sont renseignées avec les propriétés du fichier local.

Remarque : *le code ne doit appeler la méthode `FileReference.save()` qu'en réponse à un événement utilisateur, tel qu'un événement de type clic de souris ou pression de touche. Dans le cas contraire, une erreur est renvoyée. Cette restriction ne s'applique pas au contenu qui s'exécute dans le sandbox de sécurité de l'application Adobe AIR.*

La méthode `FileReference.save()` est renvoyée juste après son appel. L'objet FileReference distribue ensuite des événements pour appeler les méthodes d'écouteur à chaque étape du processus d'enregistrement de fichier.

L'objet FileReference distribue les événements suivants au cours du processus d'enregistrement de fichier :

- Evénement `select` (`Event.SELECT`) : distribué lorsque l'utilisateur indique l'emplacement et le nom du nouveau fichier à enregistrer.
- Evénement `cancel` (`Event.CANCEL`) : distribué lorsque l'utilisateur clique sur le bouton Annuler dans la boîte de dialogue.
- Evénement `open` (`Event.OPEN`) : distribué lorsque l'opération d'enregistrement commence.
- Evénement `progress` (`ProgressEvent.PROGRESS`) : distribué régulièrement pendant l'enregistrement des octets de données dans le fichier.
- Evénement `complete` (`Event.COMPLETE`) : distribué en cas de réussite de l'opération d'enregistrement.
- Evénement `ioError` (`IOErrorEvent.IO_ERROR`) : distribué si le processus d'enregistrement échoue en raison d'une erreur d'entrée/sortie lors d'une tentative d'enregistrement des données dans le fichier.

Le type d'objet transmis dans le paramètre `data` de la méthode `FileReference.save()` détermine le mode d'écriture des données dans le fichier :

- S'il s'agit d'une valeur String, les données sont enregistrées au format texte UTF-8.
- S'il s'agit d'un objet XML, elles sont écrites dans un fichier XML en conservant l'ensemble du formatage.
- S'il s'agit d'un objet ByteArray, leur contenu est écrit directement dans le fichier sans conversion.
- S'il s'agit d'un autre type d'objet, la méthode `FileReference.save()` appelle la méthode `toString()` de l'objet, puis enregistre la valeur String résultante dans un fichier texte UTF-8. S'il est impossible d'appeler la méthode `toString()` de l'objet, une erreur est renvoyée.

Si la valeur du paramètre `data` est `null`, une erreur est renvoyée.

Le code suivant étend l'exemple précédent pour la méthode `FileReference.load()`. Une fois les données lues dans le fichier, cet exemple invite l'utilisateur à entrer un nom de fichier, puis enregistre les données dans un nouveau fichier :

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.FileFilter;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.utils.ByteArray;

    public class FileReferenceExample2 extends Sprite
    {
        private var fileRef:FileReference;
        public function FileReferenceExample2()
        {
            fileRef = new FileReference();
            fileRef.addEventListener(Event.SELECT, onFileSelected);
            fileRef.addEventListener(Event.CANCEL, onCancel);
            fileRef.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
            fileRef.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
                        onSecurityError);
            var textTypeFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)",
                        "*.txt;*.rtf");
            fileRef.browse([textTypeFilter]);
        }
        public function onFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.addEventListener(Event.COMPLETE, onComplete);
            fileRef.load();
        }

        public function onProgress(evt:ProgressEvent):void
        {
            trace("Loaded " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }
        public function onCancel(evt:Event):void
        {
            trace("The browse request was canceled by the user.");
        }
        public function onComplete(evt:Event):void
        {
            trace("File was successfully loaded.");
            fileRef.removeEventListener(Event.SELECT, onFileSelected);
            fileRef.removeEventListener(ProgressEvent.PROGRESS, onProgress);
            fileRef.removeEventListener(Event.COMPLETE, onComplete);
            fileRef.removeEventListener(Event.CANCEL, onCancel);
            saveFile();
        }
        public function saveFile():void
        {
            fileRef.addEventListener(Event.SELECT, onSaveFileSelected);
            fileRef.save(fileRef.data,"NewFileName.txt");
        }
```

```
        public function onSaveFileSelected(evt:Event):void
        {
            fileRef.addEventListener(ProgressEvent.PROGRESS, onSaveProgress);
            fileRef.addEventListener(Event.COMPLETE, onSaveComplete);
            fileRef.addEventListener(Event.CANCEL, onSaveCancel);
        }

        public function onSaveProgress(evt:ProgressEvent):void
        {
            trace("Saved " + evt.bytesLoaded + " of " + evt.bytesTotal + " bytes.");
        }

        public function onSaveComplete(evt:Event):void
        {
            trace("File saved.");
            fileRef.removeEventListener(Event.SELECT, onSaveFileSelected);
            fileRef.removeEventListener(ProgressEvent.PROGRESS, onSaveProgress);
            fileRef.removeEventListener(Event.COMPLETE, onSaveComplete);
            fileRef.removeEventListener(Event.CANCEL, onSaveCancel);
        }

        public function onSaveCancel(evt:Event):void
        {
            trace("The save request was canceled by the user.");
        }

        public function onIOError(evt:IOErrorEvent):void
        {
            trace("There was an IO Error.");
        }
        public function onSecurityError(evt:Event):void
        {
            trace("There was a security error.");
        }
    }
}
```

Au terme du chargement des données à partir du fichier, le code appelle la méthode `onComplete()`. La méthode `onComplete()` supprime les écouteurs associés aux événements de chargement, puis appelle la méthode `saveFile()`. La méthode `saveFile()` appelle la méthode `FileReference.save()`. La méthode `FileReference.save()` ouvre une nouvelle boîte de dialogue dans laquelle l'utilisateur entre un nouveau nom de fichier et l'emplacement d'enregistrement de ce dernier. Les autres méthodes d'écouteur d'événement tracent le déroulement du processus d'enregistrement du fichier jusqu'à ce qu'il soit terminé.

Dans Adobe AIR, la classe FileStream propose d'autres fonctionnalités d'écriture des données dans un fichier local (voir « Lecture et écriture de fichiers » à la page 715).

Chargement de fichiers sur un serveur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour charger des fichiers sur un serveur, commencez par appeler la méthode `browse()` pour permettre à l'utilisateur de sélectionner un ou plusieurs fichiers. Après l'appel de la méthode `FileReference.upload()`, le fichier sélectionné est transféré sur le serveur. Si l'utilisateur sélectionne plusieurs fichiers à l'aide de la méthode `FileReferenceList.browse()`, Flash Player crée un tableau de fichiers sélectionnés appelé `FileReferenceList.fileList`. Vous pouvez alors utiliser la méthode `FileReference.upload()` pour charger chaque fichier individuellement.

Remarque : *l'utilisation de la méthode `FileReference.browse()` ne vous permet de charger qu'un seul fichier à la fois. Pour autoriser l'utilisateur à charger plusieurs fichiers, faites appel à la méthode `FileReferenceList.browse()`.*

Par défaut, la boîte de dialogue de sélection de fichier du système d'exploitation permet à l'utilisateur de choisir tout type de fichier sur l'ordinateur local. Les développeurs peuvent toutefois filtrer les types de fichier à l'aide de la classe FileFilter, en transmettant un tableau d'occurrences de filtres de fichiers à la méthode `browse()` :

```
var imageTypes:FileFilter = new FileFilter("Images (*.jpg, *.jpeg, *.gif, *.png)", "*.jpg;
*.jpeg; *.gif; *.png");
var textTypes:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)", "*.txt; *.rtf");
var allTypes:Array = new Array(imageTypes, textTypes);
var fileRef:FileReference = new FileReference();
fileRef.browse(allTypes);
```

Lorsque l'utilisateur a sélectionné les fichiers et cliqué sur le bouton Ouvrir dans la boîte de dialogue de sélection système, l'événement `Event.SELECT` est distribué. Si vous sélectionnez le fichier à charger à l'aide de la méthode `FileReference.browse()`, le code suivant envoie le fichier à un serveur Web :

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
try
{
    var success:Boolean = fileRef.browse();
}
catch (error:Error)
{
    trace("Unable to browse for files.");
}
function selectHandler(event:Event):void
{
    var request:URLRequest = new URLRequest("http://www.[yourdomain].com/fileUploadScript.cfm")
    try
    {
        fileRef.upload(request);
    }
    catch (error:Error)
    {
        trace("Unable to upload file.");
    }
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

Avec la méthode `FileReference.upload()`, vous pouvez envoyer des données au serveur en utilisant les propriétés `URLRequest.method` et `URLRequest.data` en vue d'envoyer des variables à l'aide de la méthode `POST` ou `GET`.

Si vous tentez de charger un fichier à l'aide de la méthode `FileReference.upload()`, les événements suivants sont distribués :

- Evénement `open` (`Event.OPEN`) : distribué lorsque l'opération de chargement commence.
- Evénement `progress` (`ProgressEvent.PROGRESS`) : distribué régulièrement pendant le chargement des octets de données du fichier.
- Evénement `complete` (`Event.COMPLETE`) : distribué en cas de réussite de l'opération de chargement.
- Evénement `httpStatus` (`HTTPStatusEvent.HTTP_STATUS`) : distribué lorsque le processus de chargement échoue en raison d'une erreur HTTP.
- Evénement `httpResponseStatus` (`HTTPStatusEvent.HTTP_RESPONSE_STATUS`) : distribué si un appel de la méthode `upload()` ou `uploadUnencoded()` tente d'accéder aux données via HTTP, et si Adobe AIR est capable de détecter et de renvoyer le code d'état de la requête.
- Evénement `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) : distribué lorsqu'une opération de chargement échoue en raison d'une violation de la sécurité.
- Evénement `uploadCompleteData` (`DataEvent.UPLOAD_COMPLETE_DATA`) : distribué après réception des données par le serveur suite à un chargement réussi.
- Evénement `ioError` (`IOErrorEvent.IO_ERROR`) : distribué si le processus de chargement échoue pour l'une des raisons suivantes :
  - Une erreur d'entrée/sortie se produit dans Flash Player pendant la lecture, l'écriture ou la transmission du fichier.
  - Le fichier SWF tente de charger un fichier sur un serveur nécessitant une authentification (un nom d'utilisateur et un mot de passe, par exemple). Au cours du chargement, Flash Player ne permet pas aux utilisateurs d'entrer des mots de passe.
  - Le paramètre `url` contient un protocole incorrect. La méthode `FileReference.upload()` doit utiliser HTTP ou HTTPS.

Flash Player n'offre pas une prise en charge complète des serveurs nécessitant une authentification. Seuls les fichiers SWF s'exécutant dans un navigateur, via le module d'extension du navigateur ou le contrôle Microsoft ActiveX®, peuvent fournir une boîte de dialogue pour inviter l'utilisateur à entrer un nom et un mot de passe d'authentification, et ce uniquement pour les téléchargements. Le transfert de fichiers échoue si le chargement est effectué à l'aide du module d'extension ou du contrôle ActiveX, ou si un chargement/téléchargement est effectué par le biais du lecteur autonome ou externe.

Pour créer un script serveur dans ColdFusion de manière à accepter les chargements de fichier en provenance de Flash Player, vous pouvez utiliser un code semblable à celui-ci :

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#"
nameconflict="OVERWRITE" />
```

Ce code ColdFusion charge le fichier envoyé par Flash Player et l'enregistre dans le même répertoire que le modèle ColdFusion, en écrasant tout fichier existant du même nom. Cet exemple présente le minimum de code nécessaire à l'acceptation du chargement d'un fichier ; ce script ne doit pas être utilisé dans un environnement de production. Ajoutez de préférence un mécanisme de validation des données pour garantir que les utilisateurs chargent uniquement des types de fichiers autorisés, telle une image plutôt qu'un script côté serveur potentiellement dangereux.

Le code ci-après présente le chargement de fichiers via PHP, avec validation des données. Le script limite le nombre de fichiers chargés dans le répertoire cible à 10, vérifie que le fichier fait moins de 200 Ko et autorise uniquement le chargement et l'enregistrement de fichiers JPEG, GIF ou PNG.

```
<?php
$MAXIMUM_FILESIZE = 1024 * 200; // 200KB
$MAXIMUM_FILE_COUNT = 10; // keep maximum 10 files on server
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE)
{
    move_uploaded_file($_FILES['Filedata']['tmp_name'],
"./temporary/".$_FILES['Filedata']['name']);
    $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
    if ($type == 1 || $type == 2 || $type == 3)
    {
        rename("./temporary/".$_FILES['Filedata']['name'],
"./images/".$_FILES['Filedata']['name']);
    }
    else
    {
        unlink("./temporary/".$_FILES['Filedata']['name']);
    }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory))
{
    array_push($files, array('./images/'.$file, filectime('./images/'.$file)));
}
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT)
{
    $files_to_delete = array_splice($files, 0, count($files) - $MAXIMUM_FILE_COUNT);
    for ($i = 0; $i < count($files_to_delete); $i++)
    {
        unlink($files_to_delete[$i][0]);
    }
}
print_r($files);
closedir($directory);

function sorter($a, $b)
{
    if ($a[1] == $b[1])
    {
        return 0;
    }
    else
    {
        return ($a[1] < $b[1]) ? -1 : 1;
    }
}
?>
```

Vous pouvez transmettre des variables supplémentaires au script de chargement à l'aide de la méthode de requête `POST` ou `GET`. Pour envoyer des variables `POST` au script de chargement, vous pouvez utiliser le code suivant :

```
var fileRef:FileReference = new FileReference();
fileRef.addEventListener(Event.SELECT, selectHandler);
fileRef.addEventListener(Event.COMPLETE, completeHandler);
fileRef.browse();
function selectHandler(event:Event):void
{
    var params:URLVariables = new URLVariables();
    params.date = new Date();
    params.ssid = "94103-1394-2345";
    var request:URLRequest = new
URLRequest("http://www.yourdomain.com/FileReferenceUpload/fileupload.cfm");
    request.method = URLRequestMethod.POST;
    request.data = params;
    fileRef.upload(request, "Custom1");
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

L'exemple précédent crée un objet URLVariables à transmettre au script côté serveur distant. Dans les versions précédentes d'ActionScript, il était possible de transmettre des variables au script de chargement en passant des valeurs dans la chaîne de requête. ActionScript 3.0 vous permet de transmettre des variables au script distant à l'aide de l'objet URLRequest. Vous pouvez ainsi transmettre des données à l'aide de la méthode `POST` ou `GET`, ce qui simplifie et rationalise le transfert de gros volumes de données. Pour spécifier si les variables sont transmises à l'aide de la méthode de requête `GET` ou `POST`, il est possible de définir la propriété `URLRequest.method` sur `URLRequestMethod.GET` ou `URLRequestMethod.POST`, respectivement.

ActionScript 3.0 vous permet de remplacer le nom de champ par défaut du fichier à charger (`Filedata`) en ajoutant un deuxième paramètre à la méthode `upload()`, comme illustré dans l'exemple précédent (dans lequel la valeur par défaut `Filedata` est remplacée par `Custom1`).

Par défaut, Flash Player ne tente pas d'effectuer un chargement de test ; vous pouvez toutefois le faire en transmettant la valeur `true` comme troisième paramètre de la méthode `upload()`. L'objectif du test est de vérifier que le chargement véritable se fera sans problème et que l'authentification du serveur, si nécessaire, réussira.

Remarque : *actuellement, le test du chargement s'effectue uniquement dans les versions Windows de Flash Player.*

Le script serveur qui gère le chargement doit attendre une requête HTTP `POST` comportant les éléments suivants :

- `Content-Type` avec la valeur `multipart/form-data`.
- `Content-Disposition` avec comme attribut `name` « `Filedata` » et comme attribut `filename` le nom du fichier d'origine. Pour spécifier un attribut `name`, transmettez une valeur pour le paramètre `uploadDataFieldName` dans la méthode `FileReference.upload()`.
- Le contenu binaire du fichier.

Voici un exemple de requête HTTP `POST` :

```
POST /handler.asp HTTP/1.1
Accept: text/*
Content-Type: multipart/form-data;
boundary=----------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
User-Agent: Shockwave Flash
Host: www.mydomain.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache

------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Filename"

sushi.jpg
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Filedata"; filename="sushi.jpg"
Content-Type: application/octet-stream

Test File
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
Content-Disposition: form-data; name="Upload"

Submit Query
------------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
(actual file data,,,)
```

L'exemple de requête HTTP `POST` suivant envoie trois variables `POST` : `api_sig`, `api_key` et `auth_token`, puis utilise la valeur de champ de données `"photo"` :

```
POST /handler.asp HTTP/1.1
Accept: text/*
Content-Type: multipart/form-data;
boundary=----------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
User-Agent: Shockwave Flash
Host: www.mydomain.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache

------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Filename"

sushi.jpg
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_sig"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_key"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="auth_token"

XXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="photo"; filename="sushi.jpg"
Content-Type: application/octet-stream

(actual file data,,,)
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Upload"

Submit Query
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7--
```

Téléchargement de fichiers à partir d'un serveur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez autoriser les utilisateurs à télécharger des fichiers à partir d'un serveur grâce à la méthode `FileReference.download()`, qui prend deux paramètres : `request` et `defaultFileName`. Le premier paramètre est l'objet URLRequest contenant l'URL du fichier à télécharger. Le second est facultatif ; il permet de spécifier un nom de fichier par défaut qui apparaîtra dans la boîte de dialogue de téléchargement. Si vous ignorez le second paramètre, `defaultFileName`, le nom de fichier utilisé est dérivé de l'URL.

Le code suivant télécharge un fichier nommé index.xml à partir du même répertoire que le fichier SWF :

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

Pour utiliser comme nom par défaut currentnews.xml au lieu d'index.xml, spécifiez le paramètre `defaultFileName`, comme illustré par le fragment de code suivant :

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

Le changement de nom du fichier peut s'avérer utile si le nom du fichier sur le serveur est peu évocateur ou généré automatiquement. Il est également judicieux de spécifier le paramètre `defaultFileName` lorsque vous téléchargez un fichier à l'aide d'un script côté serveur, au lieu d'effectuer un téléchargement direct. Par exemple, il est nécessaire de spécifier le paramètre `defaultFileName` si vous utilisez un script côté serveur qui télécharge des fichiers en fonction des variables URL qui lui sont transmises. Autrement, le nom par défaut du fichier téléchargé est le nom du script côté serveur.

Vous pouvez également envoyer des données au serveur avec l'appel de la méthode `download()` en ajoutant des paramètres à l'URL pour que le script serveur les analyse. L'extrait de code ActionScript 3.0 ci-après télécharge un document en fonction des paramètres transmis à un script ColdFusion :

```
package
{
    import flash.display.Sprite;
    import flash.net.FileReference;
    import flash.net.URLRequest;
    import flash.net.URLRequestMethod;
    import flash.net.URLVariables;

    public class DownloadFileExample extends Sprite
    {
        private var fileToDownload:FileReference;
        public function DownloadFileExample()
        {
            var request:URLRequest = new URLRequest();
            request.url = "http://www.[yourdomain].com/downloadfile.cfm";
            request.method = URLRequestMethod.GET;
            request.data = new URLVariables("id=2");
            fileToDownload = new FileReference();
            try
            {
                fileToDownload.download(request, "file2.txt");
            }
            catch (error:Error)
            {
                trace("Unable to download file.");
            }
        }
    }
}
```

Le code suivant présente le script ColdFusion download.cfm, qui télécharge l'un des deux fichiers stockés sur le serveur en fonction de la valeur d'une variable URL :

```
<cfparam name="URL.id" default="1" />
<cfswitch expression="#URL.id#">
    <cfcase value="2">
        <cfcontent type="text/plain" file="#ExpandPath('two.txt')#" deletefile="No" />
    </cfcase>
    <cfdefaultcase>
        <cfcontent type="text/plain" file="#ExpandPath('one.txt')#" deletefile="No" />
    </cfdefaultcase>
</cfswitch>
```

Classe FileReferenceList

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe FileReferenceList permet à l'utilisateur de sélectionner un ou plusieurs fichiers à charger dans un script côté serveur. Le chargement de fichiers est géré par la méthode `FileReference.upload()`, qui doit être appelée pour chaque fichier sélectionné.

Le code suivant crée deux objets FileFilter (`imageFilter` et `textFilter`) et les transmet sous forme de tableau à la méthode `FileReferenceList.browse()`. Ainsi, la boîte de dialogue du système d'exploitation propose deux types de fichiers possibles.

```
var imageFilter:FileFilter = new FileFilter("Image Files (*.jpg, *.jpeg, *.gif, *.png)",
"*.jpg; *.jpeg; *.gif; *.png");
var textFilter:FileFilter = new FileFilter("Text Files (*.txt, *.rtf)", "*.txt; *.rtf");
var fileRefList:FileReferenceList = new FileReferenceList();
try
{
    var success:Boolean = fileRefList.browse(new Array(imageFilter, textFilter));
}
catch (error:Error)
{
    trace("Unable to browse for files.");
}
```

L'utilisation de la classe FileReferenceList pour autoriser le chargement de fichiers est semblable à l'utilisation de `FileReference.browse()`, à la différence que FileReferenceList permet de sélectionner plusieurs fichiers. En cas de sélection de fichiers multiples, il est nécessaire de charger chacun des fichiers choisis à l'aide de `FileReference.upload()`, comme le montre le code suivant :

```
var fileRefList:FileReferenceList = new FileReferenceList();
fileRefList.addEventListener(Event.SELECT, selectHandler);
fileRefList.browse();

function selectHandler(event:Event):void
{
    var request:URLRequest = new URLRequest("http://www.[yourdomain].com/fileUploadScript.cfm");
    var file:FileReference;
    var files:FileReferenceList = FileReferenceList(event.target);
    var selectedFileArray:Array = files.fileList;
    for (var i:uint = 0; i < selectedFileArray.length; i++)
    {
        file = FileReference(selectedFileArray[i]);
        file.addEventListener(Event.COMPLETE, completeHandler);
        try
        {
            file.upload(request);
        }
        catch (error:Error)
        {
            trace("Unable to upload files.");
        }
    }
}
function completeHandler(event:Event):void
{
    trace("uploaded");
}
```

Comme l'événement `Event.COMPLETE` s'ajoute à chaque objet FileReference dans le tableau, Flash Player appelle la méthode `completeHandler()` à la fin du chargement de chacun des fichiers.

Utilisation de l'interface de programmation du système de fichiers AIR

Adobe AIR 1.0 et les versions ultérieures

L'API du système de fichiers Adobe AIR comprend les classes suivantes :

- Fichier
- FileMode
- FileStream

L'API du système de fichiers vous permet d'exécuter entre autres les opérations suivantes :

- Copier, créer, supprimer et déplacer des fichiers et des répertoires
- Obtenir des informations sur les fichiers et les répertoires
- Lire et écrire des fichiers

Principes de base des classes File d'AIR

Adobe AIR 1.0 et les versions ultérieures

Pour obtenir une explication rapide de l'utilisation du système de fichiers dans AIR et des exemples de code correspondants, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Building a text-file editor (Flash) (disponible en anglais uniquement)
- Développement d'un éditeur de fichier texte (Flex)
- Développement d'une application de recherche dans des répertoires (Flex)
- Lecture et écriture à partir d'un fichier de préférences XML (Flex)
- Compression de fichiers et de données (Flex)

Adobe AIR propose des classes permettant de créer et de gérer des fichiers et des dossiers, ainsi que d'y accéder. Ces classes, qui résident dans le package flash.filesystem, s'utilisent comme suit :

Classes File	Description
File	Objet File représentant le chemin d'accès à un fichier ou un répertoire. Vous utilisez un objet File pour créer un pointeur vers un fichier ou un dossier, ce qui initie une interaction avec le fichier ou le dossier.
FileMode	La classe FileMode définit des constantes chaîne qui sont utilisées dans le paramètre `fileMode` des méthodes `open()` et `openAsync()` de la classe FileStream. Le paramètre `fileMode` de ces méthodes détermine les fonctionnalités à la disposition de l'objet FileStream une fois le fichier ouvert, notamment l'écriture, la lecture, l'ajout en fin de fichier et la mise à jour.
FileStream	Objet FileStream permettant d'ouvrir des fichiers à des fins de lecture ou d'écriture. Une fois créé un objet File pointant vers un fichier nouveau ou existant, vous transmettez ce pointeur à l'objet FileStream dans le but d'ouvrir ce fichier et d'écrire ou de lire des données.

Certaines méthodes de la classe File possèdent des versions synchrone et asynchrone :

- `File.copyTo()` et `File.copyToAsync()`
- `File.deleteDirectory()` et `File.deleteDirectoryAsync()`
- `File.deleteFile()` et `File.deleteFileAsync()`
- `File.getDirectoryListing()` et `File.getDirectoryListingAsync()`
- `File.moveTo()` et `File.moveToAsync()`
- `File.moveToTrash()` et `File.moveToTrashAsync()`

En outre, les opérations FileStream s'exécutent en mode synchrone ou asynchrone selon que l'objet FileStream ouvre le fichier par appel de la méthode `open()` ou `openAsync()`.

Les versions asynchrones vous permettent d'initier des processus s'exécutant en arrière-plan et de distribuer des événements lorsqu'ils sont terminés (ou lorsqu'un événement erreur se produit). Un autre code peut s'exécuter pendant que ces processus asynchrones se déroulent en arrière-plan. Lorsque vous utilisez la version asynchrone de ces opérations, vous devez configurer des fonctions d'écouteur d'événement, à l'aide de la méthode `addEventListener()` de l'objet File ou FileStream qui appelle la fonction.

Les versions synchrones permettent d'écrire du code plus simple qui ne repose pas sur la configuration d'écouteurs d'événement. Cependant, comme aucun autre code ne peut s'exécuter pendant l'exécution d'une méthode synchrone, il est possible que des processus importants tels que l'animation et le rendu des objets d'affichage soient retardés.

Utilisation des objets File dans AIR

Adobe AIR 1.0 et les versions ultérieures

Un objet File est un pointeur vers un fichier ou un répertoire du système de fichiers.

La classe File étend la classe FileReference. La classe FileReference, qui est disponible dans Adobe® Flash® Player et AIR, représente un pointeur vers un fichier. La classe File ajoute des propriétés et des méthodes qui ne sont pas exposées dans Flash Player (dans un fichier SWF s'exécutant dans un navigateur), pour des raisons de sécurité.

Présentation de la classe File

Adobe AIR 1.0 et les versions ultérieures

La classe File permet d'effectuer les opérations suivantes :

- Obtenir le chemin d'accès à des répertoires particuliers, notamment le répertoire de l'utilisateur, le répertoire de documents de l'utilisateur, le répertoire de lancement de l'application et le répertoire d'application
- Copier des fichiers et des répertoires
- Déplacer des fichiers et des répertoires
- Supprimer des fichiers et des répertoires (ou les transférer dans la corbeille)
- Afficher la liste des fichiers et répertoires que contient un répertoire
- Créer des fichiers et des dossiers temporaires

Une fois qu'un objet File pointe vers un chemin de fichier, vous pouvez l'utiliser pour lire et écrire des données de fichier, à l'aide de la classe FileStream.

Un objet File peut pointer vers le chemin d'un fichier ou d'un répertoire qui n'existe pas encore. Vous pouvez utiliser un tel objet File pour créer un fichier ou un répertoire.

Chemin des objets File

Adobe AIR 1.0 et les versions ultérieures

Tout objet File possède deux propriétés qui définissent son chemin :

Propriété	Description
`nativePath`	Indique le chemin d'accès à un fichier en fonction de la plate-forme. Par exemple, sous Windows, un chemin se présente sous la forme « c:\Sample directory\test.txt » alors que sous Mac OS, il correspond à « /Sample directory/test.txt ». La propriété `nativePath` utilise la barre oblique inverse (\) comme séparateur de répertoires sous Windows et la barre oblique normale (/) sous Mac OS et Linux.
`url`	Cette propriété peut utiliser le modèle d'URL file pour pointer vers un fichier. Par exemple, sous Windows, un chemin se présente sous la forme « file:///c:/Sample%20directory/test.txt » alors que sous Mac OS, il correspond à « file:///Sample%20directory/test.txt ». Outre `file`, le moteur d'exécution propose d'autres modèles d'URL particuliers, qui sont décrits à la section « Modèles d'URL AIR pris en charge » à la page 702.

La classe File comprend des propriétés statiques de pointage vers les répertoires standard sous Mac, Windows et Linux. Parmi ces propriétés figurent :

- `File.applicationStorageDirectory` : répertoire de stockage spécifique à chaque application AIR installée. Ce répertoire est adapté au stockage des actifs d'applications dynamiques et des préférences utilisateur. Il est recommandé de stocker les volumes élevés de données dans un autre répertoire.

  Sous Android et iOS, le répertoire de stockage de l'application est supprimé lorsque l'utilisateur désinstalle l'application ou lorsqu'il décide d'effacer les données de l'application. Ce n'est toutefois pas le cas sur d'autres plates-formes.

- `File.applicationDirectory` : répertoire dans lequel est installée l'application (et qui contient tout actif installé). Sur certains systèmes d'exploitation, l'application est stockée dans un fichier de package unique, plutôt que dans un répertoire physique. Dans ce cas de figure, il est parfois impossible d'accéder au contenu via le chemin natif. Le répertoire d'application est disponible en lecture seule.
- `File.desktopDirectory` : répertoire du poste de travail de l'utilisateur. Si une plate-forme ne définit pas de répertoire de poste de travail, un autre emplacement du système de fichiers est alors utilisé.
- `File.documentsDirectory` : répertoire de documents de l'utilisateur. Si une plate-forme ne définit pas de répertoire de documents, un autre emplacement du système de fichiers est alors utilisé.
- `File.userDirectory` : répertoire de l'utilisateur. Si une plate-forme ne définit pas de répertoire utilisateur, un autre emplacement du système de fichiers est alors utilisé.

Remarque : *si une plate-forme ne définit pas d'emplacement standard pour les répertoires de poste de travail, de documents ou d'utilisateur,* `File.documentsDirectory`*,* `File.desktopDirectory` *et* `File.userDirectory` *peuvent pointer vers le même répertoire.*

La valeur de ces propriétés varie selon le système d'exploitation. Par exemple, le chemin natif vers le répertoire du poste de travail de l'utilisateur est différent sur Mac et Windows. La propriété `File.desktopDirectory` pointe néanmoins vers un chemin de répertoire approprié sur chaque plate-forme. Pour développer des applications qui fonctionnent correctement sur toutes les plateformes, utilisez ces propriétés comme base pour référencer d'autres fichiers et répertoires dont se sert l'application. Utilisez ensuite la méthode `resolvePath()` pour affiner le chemin. Par exemple, le code suivant pointe vers le fichier preferences.xml dans le répertoire de stockage de l'application :

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

Bien que la classe File vous permette de pointer vers un chemin de fichier déterminé, les applications risquent alors de ne pas fonctionner sur toutes les plates-formes. Par exemple, le chemin C:\Documents and Settings\joe\ ne fonctionne que sous Windows. C'est pourquoi il est préférable d'utiliser les propriétés statiques de la classe File, telles que `File.documentsDirectory`.

Emplacements de répertoire standard

Plate-forme	Type de répertoire	Emplacement de système de fichiers typique
Android	Application	`/data/data/`
	Stockage d'application	`/data/data/air.IDApplication/NomFichier/Local Store`
	Cache	/data/data/applicationID/cache
	Bureau	`/mnt/sdcard`
	Documents	`/mnt/sdcard`
	Temporaire	`/data/data/IDApplication/cache/FlashTmp.randomString`
	Utilisateur	`/mnt/sdcard`
iOS	Application	`/var/mobile/Applications/uid/filename.app`
	Stockage d'application	`/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store`
	Cache	/var/mobile/Applications/uid/Library/Caches
	Bureau	non accessible
	Documents	`/var/mobile/Applications/uid/Documents`
	Temporaire	`/private/var/mobile/Applications/uid/tmp/FlashTmpNNN`
	Utilisateur	non accessible
Linux	Application	`/opt/NomFichier/share`
	Stockage d'application	`/home/NomUtilisateur/.appdata/IDApplicationID/Local Store`
	Bureau	`/home/NomUtilisateur/Desktop`
	Documents	`/home/NomUtilisateur/Documents`
	Temporaire	`/tmp/FlashTmp.randomString`
	Utilisateur	`/home/NomUtilisateur`
Mac	Application	`/Applications/NomFichier.app/Contents/Resources`
	Stockage d'application	`/Utilisateurs/`***`nomUtilisateur`***`/Bibliothèque/Préférences/`***`applicationid`***`/Local Store` (AIR 3.2 et versions ultérieures) ***`chemin`***`/Bibliothèque/Application Support/`***`applicationid`***`/Local Store` (AIR 3.3 et versions ultérieures), où le chemin est soit `/Utilisateurs/`***`nomUtilisateur`***`/Bibliothèque/Containers/`***`bundle-id`***`/Data` (environnement de sandbox), soit `/Utilisateurs/nomUtilisateur` (exécution en dehors d'un environnement de sandbox)
	Cache	/Users/NomUtilisateur/Library/Caches
	Bureau	`/Utilisateurs/`***`nomUtilisateur`***`/Desktop`
	Documents	`/Utilisateurs/`***`nomUtilisateur`***`/Documents`
	Temporaire	`/private/var/folders/JY/randomString/TemporaryItems/FlashTmp`
	Utilisateur	`/Utilisateurs/`***`nomUtilisateur`***

Plate-forme	Type de répertoire	Emplacement de système de fichiers typique
Windows	Application	`C:\Program Files\NomFichier`
	Stockage d'application	`C:\Documents and settings\NomUtilisateur\ApplicationData\IDApplication\Local Store`
	Cache	C:\Documents and settings\NomUtilisateur\Local Settings\Temp
	Bureau	`C:\Documents and settings\NomUtilisateur\Desktop`
	Documents	`C:\Documents and settings\NomUtilisateur\Mes documents`
	Temporaire	`C:\Documents and settings\NomUtilisateur\Local Settings\Temp\randomString.tmp`
	Utilisateur	`C:\Documents and settings\NomUtilisateur`

Les chemins natifs de ces répertoires varient en fonction du système d'exploitation et de la configuration de l'ordinateur. Les chemins indiqués dans ce tableau représentent des exemples typiques. Pointez toujours vers ces répertoires par le biais des propriétés statiques appropriées de la classe File, afin que l'application fonctionne correctement quelle que soit la plate-forme. Dans une application AIR réelle, les valeurs des variables `IDApplication` et `NomFichier` indiquées dans le tableau sont extraites du descripteur d'application. Si vous stipulez un ID d'éditeur dans le descripteur d'application, il est ajouté à la fin de l'ID d'application dans ces chemins. La valeur de `NomUtilisateur` correspond au nom du compte de l'utilisateur qui installe l'application.

Pointage d'un objet File vers un répertoire

Adobe AIR 1.0 et les versions ultérieures

Vous disposez de plusieurs méthodes pour configurer un objet File afin qu'il pointe vers un répertoire.

Pointage vers le répertoire d'accueil de l'utilisateur

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer un objet File vers le répertoire d'accueil de l'utilisateur. Le code suivant configure l'objet File afin qu'il pointe vers le sous-répertoire AIR Test du répertoire d'accueil :

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

Pointage vers le répertoire de documents de l'utilisateur

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer un objet File vers le répertoire de documents de l'utilisateur. Le code suivant configure un objet File afin qu'il pointe vers le sous-répertoire AIR Test du répertoire de documents :

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

Pointage vers le répertoire du poste de travail

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer un objet File vers le poste de travail. Le code suivant configure un objet File afin qu'il pointe vers le sous-répertoire AIR Test du poste de travail :

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

Pointage vers le répertoire de stockage d'une application

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer un objet File vers le répertoire de stockage d'une application. A toute application AIR est associé un chemin unique qui définit son répertoire de stockage. Ce répertoire est spécifique à chaque application et utilisateur. Vous pouvez y stocker des données spécifiques à l'application et à l'utilisateur, notamment des données utilisateur ou des fichiers de préférences. Par exemple, le code suivant pointe un objet File vers un fichier de préférences, prefs.xml, qui réside dans le répertoire de stockage de l'application :

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

L'emplacement du répertoire de stockage de l'application est généralement déterminé par le nom d'utilisateur et l'ID d'application. Les emplacements suivants au sein du système de fichiers sont indiqués ci-après à des fins de débogage d'application. Utilisez toujours la propriété `File.applicationStorage` ou le modèle d'URI `app-storage:` pour résoudre les fichiers dans ce répertoire :

- Sous Mac OS : dépend de la version AIR :

  AIR 3.2 et versions antérieures : `/Utilisateurs/nom utilisateur/Bibliothèque/Préférences/`*`applicationID`*`/Local Store/`

  AIR 3.3 et versions ultérieures : *`chemin`*`/Bibliothèque/Application Support/`*`applicationID`*`/Local Store`, où *`chemin`* est soit `/Utilisateurs/`*`nomUtilisateur`*`/Bibliothèque/Containers/`*`bundle-id`*`/Data` (environnement de sandbox), soit `/Utilisateurs/`*`nomUtilisateur`* (exécution en dehors d'un environnement de sandbox)

  Par exemple (AIR 3.2) :

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- Sous Windows, dans le répertoire Documents and Settings, dans :

  C:\Documents and Settings\nom d'utilisateur`\Application Data\`*IDapplication*`\Local Store\`

  Exemple :

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- Sous Linux, dans :

  `/home/`*nom d'utilisateur*`/.appdata/`*IDapplication*`/Local Store/`

  Exemple :

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- Sur Android, dans :

  /data/data/`IDPackageAndroid/IDapplication/Local Store`

  Exemple :

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

Remarque : *si une application possède un identifiant d'éditeur, ce dernier figure également dans le chemin du répertoire de stockage de l'application.*

L'URL (et la propriété `url`) d'un objet File créé à l'aide de `File.applicationStorageDirectory` utilise le modèle d'URL `app-storage` (voir « Modèles d'URL AIR pris en charge » à la page 702), comme l'illustre l'exemple suivant :

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

Pointage vers le répertoire de l'application

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer un objet File vers le répertoire dans lequel une application est installée, ou répertoire d'application. Vous pouvez référencer ce répertoire à l'aide de la propriété `File.applicationDirectory`. Il permet d'examiner le fichier descripteur d'application ou d'autres ressources installées avec l'application. Par exemple, le code suivant pointe un objet File vers le répertoire *images* du répertoire d'application :

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

L'URL (et la propriété `url`) d'un objet File créé à l'aide de `File.applicationDirectory` utilise le modèle d'URL `app` (voir « Modèles d'URL AIR pris en charge » à la page 702), comme l'illustre l'exemple suivant :

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

Remarque : *sur Android, il est impossible d'accéder aux fichiers du package d'application par le biais de `nativePath`. La propriété `nativePath` est une chaîne vide. Accédez toujours aux fichiers que contient le répertoire de l'application par le biais de l'URL, plutôt que d'un chemin natif.*

Pointage vers le répertoire du cache

Adobe AIR 3.6 et les versions ultérieures

Vous pouvez pointer un objet File vers le répertoire temporaire ou de cache du système d'exploitation en utilisant la propriété `File.cacheDirectory`. Ce répertoire contient des fichiers temporaires qui ne sont pas nécessaires à l'exécution de l'application et dont la suppression ne cause pas de problèmes ni de pertes de données pour l'utilisateur.

Dans la plupart des systèmes d'exploitation, le répertoire de cache est un répertoire temporaire. Sous iOS, le répertoire de cache correspond au répertoire Caches de la bibliothèque d'applications. Les fichiers de ce répertoire ne sont pas sauvegardés sur le stockage en ligne et peuvent être supprimés par le système d'exploitation si l'espace de stockage disponible sur le périphérique est insuffisant. Pour plus d'informations, voir « Contrôle de la sauvegarde et de la mise en cache des fichiers » à la page 702.

Pointage vers la racine du système de fichiers

Adobe AIR 1.0 et les versions ultérieures

La méthode `File.getRootDirectories()` répertorie tous les volumes racine, tels que C: et les volumes montés, sur un ordinateur Windows. Sous Mac et Linux, cette méthode renvoie le répertoire racine unique de l'ordinateur (le répertoire « / »). La méthode `StorageVolumeInfo.getStorageVolumes()` propose des informations plus détaillées sur les volumes de stockage montés (voir « Utilisation des volumes de stockage » à la page 713).

Remarque : *il est impossible d'accéder en lecture à la racine du système de fichiers sur Android. La méthode renvoie un objet File qui pointe vers le répertoire par le biais du chemin natif « / », mais les propriétés de cet objet n'ont pas de valeur précise. Par exemple, `spaceAvailable` correspond toujours à 0.*

Pointage vers un répertoire explicite

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez pointer l'objet File vers un répertoire explicite en définissant sa propriété `nativePath`, comme l'illustre l'exemple ci-dessous (sous Windows) :

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

Important : utiliser cette technique pour pointer vers un chemin explicite risque de générer un code qui ne fonctionne pas sur toutes les plates-formes. Ainsi, l'exemple précédent ne fonctionne que sous Windows. Les propriétés statiques de l'objet File, telles que `File.applicationStorageDirectory`, permettent de localiser un répertoire qui fonctionne sur toutes les plates-formes. Utilisez ensuite la méthode `resolvePath()` (voir la section suivante) pour accéder à un chemin relatif.

Navigation vers des chemins relatifs

Adobe AIR 1.0 et les versions ultérieures

La méthode `resolvePath()` permet d'obtenir un chemin relatif à un autre chemin donné. Par exemple, le code suivant configure un objet File afin qu'il pointe vers le sous-répertoire « AIR Test » du répertoire d'accueil de l'utilisateur :

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

En outre, la propriété `url` d'un objet File permet de pointer celui-ci vers un répertoire basé sur une chaîne d'URL, comme illustré ci-dessous :

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

Pour plus d'informations, voir la section « Modification de chemins de fichier » à la page 702.

Sélection d'un répertoire par l'utilisateur

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend la méthode `browseForDirectory()`, qui présente une boîte de dialogue système dans laquelle l'utilisateur peut sélectionner un répertoire à affecter à l'objet. La méthode `browseForDirectory()` est asynchrone. L'objet File distribue un événement `select` si l'utilisateur sélectionne un répertoire et clique sur le bouton Ouvrir, ou un événement `cancel` si l'utilisateur clique sur le bouton Annuler.

Par exemple, le code suivant permet à l'utilisateur de sélectionner un répertoire et renvoie le chemin de celui-ci une fois la sélection effectuée :

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

Remarque : *sur Android, la méthode `browseForDirectory()` n'est pas prise en charge. Appeler cette méthode n'a pas d'effet et un événement d'annulation est immédiatement distribué. Pour que les utilisateurs puissent sélectionner un répertoire, utilisez plutôt une boîte de dialogue personnalisée définie par l'application.*

Pointage vers le répertoire d'appel de l'application

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez déterminer l'emplacement du répertoire à partir duquel une application est appelée en consultant la propriété `currentDirectory` de l'objet InvokeEvent distribué lors de l'appel de l'application. Pour plus d'informations, voir la section « Capture des arguments de ligne de commande » à la page 912.

Pointage d'un objet File vers un fichier

Adobe AIR 1.0 et les versions ultérieures

Il existe plusieurs manières de définir le fichier vers lequel pointe un objet File.

Pointage vers un chemin de fichier explicite

Adobe AIR 1.0 et les versions ultérieures

Important : pointer vers un chemin explicite risque de générer un code qui ne fonctionne pas sur toutes les plates-formes. Ainsi, le chemin C:/foo.txt ne fonctionne que sous Windows. Les propriétés statiques de l'objet File, telles que `File.applicationStorageDirectory`, permettent de localiser un répertoire qui fonctionne sur toutes les plates-formes. Utilisez ensuite la méthode `resolvePath()` (voir « Modification de chemins de fichier » à la page 702) pour accéder à un chemin relatif.

La propriété `url` d'un objet File permet de pointer celui-ci vers un fichier ou un répertoire basé sur une chaîne d'URL, comme illustré ci-dessous :

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

Vous pouvez aussi transmettre l'URL à la fonction constructeur `File()`, comme illustré ci-dessous :

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

La propriété `url` renvoie systématiquement la version URI de l'URL (les espaces sont remplacés par `"%20`, par exemple) :

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

Vous pouvez aussi utiliser la propriété `nativePath` d'un objet File pour définir un chemin explicite. Par exemple, le code suivant, exécuté sur un ordinateur Windows, définit un objet File sur le fichier test.txt du sous-répertoire AIR Test du lecteur C: :

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

Vous pouvez aussi transmettre ce chemin à la fonction constructeur `File()`, comme illustré ci-dessous :

```
var file:File = new File("C:/AIR Test/test.txt");
```

Utilisez la barre oblique (/) comme délimiteur de chemin avec la propriété `nativePath`. Sous Windows, vous pouvez également utiliser la barre oblique inverse (\ ), mais l'application ne fonctionnera pas sur d'autres plateformes.

Pour plus d'informations, voir la section « Modification de chemins de fichier » à la page 702.

Enumération des fichiers d'un répertoire

Adobe AIR 1.0 et les versions ultérieures

La méthode `getDirectoryListing()` d'un objet File permet d'obtenir un tableau d'objets File pointant vers les fichiers et sous-répertoires situés au niveau racine d'un répertoire. Pour plus d'informations, voir la section « Enumération de répertoires » à la page 709.

Sélection d'un fichier par l'utilisateur

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les méthodes suivantes, qui présentent une boîte de dialogue système dans laquelle l'utilisateur peut sélectionner un fichier à affecter à l'objet :

- `browseForOpen()`
- `browseForSave()`
- `browseForOpenMultiple()`

Toutes ces méthodes sont asynchrones. Les méthodes `browseForOpen()` et `browseForSave()` distribuent l'événement select lorsque l'utilisateur sélectionne un fichier (ou, dans le cas de browseForSave(), un chemin cible). Avec les méthodes `browseForOpen()` et `browseForSave()`, lors de la sélection, l'objet File pointe vers les fichiers sélectionnés. La méthode `browseForOpenMultiple`() distribue un événement `selectMultiple` lorsque l'utilisateur sélectionne des fichiers. L'événement `selectMultiple` est de type FileListEvent et possède une propriété `files`, c'est-à-dire un tableau d'objets File (pointant vers les fichiers sélectionnés).

Par exemple, le code suivant présente une boîte de dialogue d'ouverture de fichier à l'utilisateur, lui permettant ainsi de sélectionner un fichier :

```
 var fileToOpen:File = File.documentsDirectory;
selectTextFile(fileToOpen);

function selectTextFile(root:File):void
{
    var txtFilter:FileFilter = new FileFilter("Text", "*.as;*.css;*.html;*.txt;*.xml");
    root.browseForOpen("Open", [txtFilter]);
    root.addEventListener(Event.SELECT, fileSelected);
}

function fileSelected(event:Event):void
{
    trace(fileToOpen.nativePath);
}
```

Si une autre boîte de dialogue de navigation est ouverte dans l'application lors de l'appel d'une méthode browse, le moteur d'exécution renvoie une exception Error.

Remarque : *sur Android, les méthodes `browseForOpen()` et `browseForOpenMultiple()` permettent de sélectionner des fichiers d'image, vidéo et audio uniquement. La boîte de dialogue browseForSave() affiche elle aussi les fichiers multimédias uniquement, même si l'utilisateur peut saisir un nom de fichier arbitraire. Pour ouvrir et enregistrer des fichiers d'un autre type, envisagez l'utilisation de boîtes de dialogue personnalisées au lieu de ces méthodes.*

Modification de chemins de fichier

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez également modifier le chemin d'un objet File existant en appelant la méthode `resolvePath()` ou en intervenant sur la propriété `nativePath` ou `url` de l'objet, comme l'illustrent les exemples suivants (sous Windows) :

```
var file1:File = File.documentsDirectory;
file1 = file1.resolvePath("AIR Test");
trace(file1.nativePath); // C:\Documents and Settings\userName\My Documents\AIR Test
var file2:File = File.documentsDirectory;
file2 = file2.resolvePath("..");
trace(file2.nativePath); // C:\Documents and Settings\userName
var file3:File = File.documentsDirectory;
file3.nativePath += "/subdirectory";
trace(file3.nativePath); // C:\Documents and Settings\userName\My Documents\subdirectory
var file4:File = new File();
file4.url = "file:///c:/AIR Test/test.txt";
trace(file4.nativePath); // C:\AIR Test\test.txt
```

Si vous faites appel à la propriété `nativePath`, utilisez la barre oblique (/) comme caractère de séparation des répertoires. Sous Windows, vous pouvez également utiliser la barre oblique (\), mais évitez de le faire, sous peine de générer un code qui ne fonctionne pas sur toutes les plates-formes.

Modèles d'URL AIR pris en charge

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous définissez la propriété `url` d'un objet File dans AIR, vous pouvez utiliser les modèles d'URL suivants :

Modèle d'URL	Description
file	Permet de spécifier un chemin relatif à la racine du système de fichiers. Exemple : `file:///c:/AIR Test/test.txt` La norme URL spécifie qu'un modèle d'URL file se présente comme suit : `file://<hôte>/<chemin>`. `<hôte>` peut correspondre à la chaîne vide, ce qui signifie « la machine à partir de laquelle l'URL est interprétée ». C'est pourquoi les modèles d'URL file comportent souvent trois barres obliques (///).
app	Permet de spécifier un chemin relatif au répertoire racine de l'application installée (le répertoire contenant le fichier application.xml de l'application installée). Par exemple, le chemin suivant pointe vers le sous-répertoire images du répertoire de l'application installée : `app:/images`
app-storage	Permet de spécifier un chemin relatif au répertoire de stockage de l'application. Pour chaque application installée, AIR définit un répertoire de stockage d'application unique. C'est un emplacement pratique auquel stocker des données spécifiques à l'application concernée. Par exemple, le chemin suivant pointe vers le fichier prefs.xml, qui réside dans le sous-répertoire settings du répertoire de stockage de l'application : `app-storage:/settings/prefs.xml`

Contrôle de la sauvegarde et de la mise en cache des fichiers

Adobe AIR 3.6 et les versions ultérieures, iOS et OS X uniquement

Certains systèmes d'exploitation, notamment iOS et Mac OS X, offrent aux utilisateurs la possibilité de sauvegarder automatiquement les fichiers d'application sur un serveur de stockage distant. En outre, iOS présente des restrictions concernant la sauvegarde des fichiers et l'emplacement de stockage des différents types de fichier.

Les points suivants présentent de façon succincte comment rester en conformité avec les directives d'Apple concernant la sauvegarde et le stockage des fichiers. Pour plus d'informations, voir les sections suivantes.

- Pour spécifier qu'un fichier n'a pas besoin d'être sauvegardé et (iOS uniquement) peut être supprimé par le système d'exploitation si le périphérique manque d'espace de stockage, enregistrez le ficher dans le répertoire de cache (`File.cacheDirectory`). Il s'agit de l'emplacement de stockage par défaut sous iOS, et il doit être utilisé pour la plupart des fichiers qui peuvent être régénérés ou retéléchargés.
- Pour spécifier qu'un fichier n'a pas besoin d'être sauvegardé, mais qu'il ne doit pas être supprimé par le système d'exploitation, enregistrez-le dans l'un des répertoires de la bibliothèque d'applications, comme le répertoire de stockage des applications (`File.applicationStorageDirectory`) ou le répertoire de documents (`File.documentsDirectory`). Définissez la propriété `preventBackup` de l'objet File sur `true`. Apple demande de procéder de cette manière pour tout contenu qu'il est possible de générer ou télécharger à nouveau, mais qui est nécessaire au bon fonctionnement de votre application lors d'une utilisation hors ligne.

Spécification des fichiers à sauvegarder

Afin d'économiser l'espace de sauvegarde et de réduire le trafic réseau, les directives d'Apple pour les applications iOS et Mac spécifient que seuls les fichiers contenant des données saisies par l'utilisateur ou des données qu'il n'est pas possible de générer ou de télécharger à nouveau doivent être marqués comme « à sauvegarder ».

Par défaut, tous les fichiers des dossiers de la bibliothèque d'applications sont sauvegardés. Sous Mac OS X, il s'agit du répertoire de stockage des applications. Sous iOS, cela inclut le répertoire de stockage de l'application, le répertoire d'application, le répertoire Bureau, le répertoire de documents et le répertoire de l'utilisateur (car ces répertoires sont mappés sur les dossiers de la bibliothèque d'applications sur iOS). Par conséquent, tous les fichiers de ces répertoires sont sauvegardés par défaut sur serveur de stockage.

Si vous enregistrez dans l'un de ces emplacements un fichier que votre application peut recréer, il est recommandé de lui appliquer un marqueur de sorte que le système d'exploitation ne le sauvegarde pas. Pour indiquer qu'un fichier ne doit pas être sauvegardé, définissez la propriété `preventBackup` de l'objet File sur `true`.

Remarque : sous iOS, tout fichier se trouvant dans l'un des dossiers de la bibliothèque d'applications (même si sa propriété `preventBackup` a la valeur `true`) est marqué comme un fichier persistant que le système d'exploitation ne doit pas supprimer.

Contrôle de la mise en cache et de la suppression des fichiers

Les directives d'Apple concernant les applications iOS spécifient que, dans la mesure du possible, tout contenu qu'il est possible de régénérer doit pouvoir être supprimé par le système d'exploitation si l'espace de stockage est insuffisant sur le périphérique.

Sous iOS, les fichiers des dossiers de la bibliothèque d'applications (comme le répertoire de stockage de l'application ou le répertoire de documents) sont marqués comme permanents et ne sont pas supprimés par le système d'exploitation.

Enregistrez les fichiers que l'application peut régénérer et qui peuvent être supprimés sans risque si l'espace de stockage est insuffisant dans le répertoire de cache de l'application. Pour accéder au répertoire de cache, utilisez la propriété statique `File.cacheDirectory`.

Sous iOS, le répertoire de cache correspond au répertoire de cache de l'application (<Racine de l'application>/Library/Caches). Sous d'autres systèmes d'exploitation, ce répertoire est mappé sur un répertoire comparable. Par exemple, sous Mac OS X, il est également mappé sur le répertoire Caches de la bibliothèque d'applications. Sous Android, le répertoire de cache est mappé sur le répertoire de cache de l'application. Sous Windows, le répertoire de cache est mappé sur le répertoire temp du système d'exploitation. Sous Android et sous Windows, il s'agit du même répertoire que celui auquel accède un appel aux méthodes `createTempDirectory()` et `createTempFile()` de la classe File.

Détermination du chemin relatif entre deux fichiers

Adobe AIR 1.0 et les versions ultérieures

La méthode `getRelativePath()` vous permet de déterminer le chemin relatif entre deux fichiers :

```
var file1:File = File.documentsDirectory.resolvePath("AIR Test");
var file2:File = File.documentsDirectory
file2 = file2.resolvePath("AIR Test/bob/test.txt");

trace(file1.getRelativePath(file2)); // bob/test.txt
```

Le deuxième paramètre de la méthode `getRelativePath()`, `useDotDot`, permet le renvoi de la syntaxe `..` dans les résultats, pour représenter les répertoires parent :

```
var file1:File = File.documentsDirectory;
file1 = file1.resolvePath("AIR Test");
var file2:File = File.documentsDirectory;
file2 = file2.resolvePath("AIR Test/bob/test.txt");
var file3:File = File.documentsDirectory;
file3 = file3.resolvePath("AIR Test/susan/test.txt");

trace(file2.getRelativePath(file1, true)); // ../..
trace(file3.getRelativePath(file2, true)); // ../../bob/test.txt
```

Obtention des versions canoniques des noms de fichier

Adobe AIR 1.0 et les versions ultérieures

Les noms de fichier et de chemin ne respectent pas la casse sous Windows et Mac OS. Dans l'exemple suivant, deux objets File pointent vers un même fichier :

```
File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

Les noms de document et de répertoire, en revanche, respectent la casse. Ainsi, l'exemple suivant considère comme acquis qu'il existe un dossier appelé AIR Test dans le répertoire documents :

```
var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

La méthode `canonicalize()` convertit l'objet `nativePath` afin qu'il utilise la combinaison correcte de majuscules et de minuscules dans le nom de fichier ou de répertoire. Sur les systèmes de fichiers sensibles à la casse (tels que Linux), lorsque plusieurs noms de fichier ne diffèrent que par la casse, la méthode `canonicalize()` ajuste le chemin de sorte qu'il corresponde au premier fichier détecté (dans l'ordre déterminé par le système de fichiers).

Vous pouvez aussi utiliser la méthode `canonicalize()` pour convertir les noms de fichier courts (« 8.3 ») en noms longs sous Windows, comme illustré ci-dessous :

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

Utilisation de packages et de liens symboliques

Adobe AIR 1.0 et les versions ultérieures

Divers systèmes d'exploitation prennent en charge les fichiers de package et les fichiers de lien symbolique :

Packages—Sous Mac OS, les répertoires peuvent être désignés comme packages et apparaissent dans le Finder sous la forme d'un fichier unique plutôt que d'un répertoire.

Liens symboliques—Mac OS, Linux, et Windows Vista prennent en charge les liens symboliques. Les liens symboliques permettent à un fichier de pointer vers un autre fichier ou répertoire du disque. Bien que similaires aux alias, les liens symboliques diffèrent toutefois de ceux-ci. Un alias est systématiquement identifié en tant que fichier (plutôt que répertoire). La lecture d'un alias (ou raccourci) ou l'écriture dans celui-ci n'a aucune incidence sur le fichier ou le répertoire d'origine vers lequel il pointe. En revanche, un lien symbolique se comporte exactement comme le fichier ou le répertoire vers lequel il pointe. Il peut être identifié en tant que fichier ou répertoire. La lecture d'un lien symbolique ou l'écriture dans celui-ci affecte le fichier ou le répertoire vers lequel il pointe, pas le lien symbolique lui-même. De plus, sous Windows, la propriété `isSymbolicLink` d'un objet File référençant un point de jonction (utilisé dans le système de fichiers NTFS) est définie sur `true`.

La classe File comprend les propriétés `isPackage` et `isSymbolicLink` qui permettent de vérifier si un objet File référence un package ou un lien symbolique.

Le code suivant effectue une itération sur le répertoire du poste de travail de l'utilisateur, répertoriant les sous-répertoires qui ne sont *pas* des packages :

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (desktopNodes[i].isDirectory && !desktopNodes[i].isPackage)
    {
        trace(desktopNodes[i].name);
    }
}
```

Le code suivant effectue une itération sur le répertoire du poste de travail de l'utilisateur, répertoriant les fichiers et répertoires qui ne sont *pas* des liens symboliques :

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

La méthode `canonicalize()` modifie le chemin d'un lien symbolique de sorte qu'il pointe vers le fichier ou le répertoire auquel le fichier ou le répertoire fait référence. Le code suivant effectue une itération sur le répertoire du poste de travail de l'utilisateur et identifie les chemins référencés par des fichiers qui sont des liens symboliques :

```
 var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (desktopNodes[i].isSymbolicLink)
    {
        var linkNode:File = desktopNodes[i] as File;
        linkNode.canonicalize();
        trace(linkNode.nativePath);
    }
}
```

Détermination de l'espace disponible sur un volume

Adobe AIR 1.0 et les versions ultérieures

La propriété `spaceAvailable` d'un objet File représente l'espace disponible utilisable, en octets, à l'emplacement de l'objet. Par exemple, le code suivant vérifie l'espace disponible dans le répertoire de stockage d'application :

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

Si l'objet File référence un répertoire, la propriété `spaceAvailable` indique l'espace que peuvent utiliser les fichiers dans le répertoire. Si l'objet File référence un fichier, la propriété `spaceAvailable` indique l'espace maximal que peut occuper le fichier. Si l'emplacement de fichier n'existe pas, la propriété `spaceAvailable` est définie sur 0. Si l'objet File référence un lien symbolique, la propriété `spaceAvailable` est définie sur l'espace disponible à l'emplacement vers lequel pointe le lien symbolique.

En règle générale, l'espace disponible pour un répertoire ou un fichier correspond à l'espace disponible sur le volume contenant ce répertoire ou fichier. Cependant, l'espace disponible peut tenir compte de quotas et de limites par répertoire.

Lors de l'ajout d'un fichier ou d'un répertoire à un volume, l'espace nécessaire est généralement supérieur à la taille réelle du fichier ou à la taille du contenu du répertoire. Il se peut, par exemple, que le système d'exploitation requiert de l'espace supplémentaire pour stocker des informations d'index. Les secteurs de disque requis utilisent peut-être de l'espace en plus. En outre, l'espace disponible change dynamiquement. Il est donc impossible d'allouer tout l'espace indiqué au stockage des fichiers. Pour plus d'informations sur l'écriture dans le système de fichiers, voir la section « Lecture et écriture de fichiers » à la page 715.

La méthode `StorageVolumeInfo.getStorageVolumes()` propose des informations plus détaillées sur les volumes de stockage montés (voir « Utilisation des volumes de stockage » à la page 713).

Ouverture de fichiers dans l'application système définie par défaut

Adobe AIR 2 et ultérieur

AIR 2 permet d'ouvrir un fichier dans l'application enregistrée à cet effet par le système d'exploitation. Une application AIR peut ainsi ouvrir un fichier DOC dans l'application enregistrée à cet effet. Faites appel à la méthode `openWithDefaultApplication()` d'un objet File pour ouvrir le fichier. Le code suivant ouvre par exemple le fichier test.doc sur le bureau de l'utilisateur dans l'application associée par défaut aux fichiers DOC :

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

Remarque : *sous Linux, le type MIME du fichier, plutôt que son extension, détermine l'application associée par défaut à un fichier.*

Le code suivant permet à l'utilisateur d'accéder à un fichier MP3 et de l'ouvrir dans l'application associée par défaut à la lecture des fichiers MP3 :

```
var file:File = File.documentsDirectory;
var mp3Filter:FileFilter = new FileFilter("MP3 Files", "*.mp3");
file.browseForOpen("Open", [mp3Filter]);
file.addEventListener(Event.SELECT, fileSelected);

function fileSelected(e:Event):void
{
    file.openWithDefaultApplication();
}
```

Il est impossible d'utiliser la méthode `openWithDefaultApplication()` pour les fichiers qui résident dans le répertoire de l'application.

AIR vous interdit d'utiliser la méthode `openWithDefaultApplication()` pour ouvrir certains fichiers. Sous Windows, AIR vous interdit d'ouvrir les fichiers de type EXE ou BAT, par exemple. Sous Mac OS et Linux, AIR vous empêche d'ouvrir les fichiers qui s'exécutent dans une application spécifique (y compris Terminal et AppletLauncher sous Mac OS, ainsi que csh, bash ou ruby sous Linux). Tenter d'ouvrir l'un de ces fichiers par le biais de la méthode `openWithDefaultApplication()` renvoie une exception. Pour obtenir la liste complète des types de fichiers concernés, voir la rubrique du guide de référence du langage consacrée à la méthode `File.openWithDefaultApplication()`.

Remarque : *cette limitation n'existe pas pour une application AIR installée à l'aide d'un programme d'installation natif (application de bureau étendue).*

Obtention d'informations sur le système de fichiers

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les propriétés statiques suivantes, qui fournissent des renseignements utiles sur le système de fichiers :

Propriété	Description
`File.lineEnding`	Séquence de caractères de fin de ligne utilisée par le système d'exploitation hôte. Sous Mac OS et Linux, il s'agit du caractère de nouvelle ligne. Sous Windows, il s'agit du retour chariot suivi du caractère de nouvelle ligne.
`File.separator`	Séparateur d'élément de chemin utilisé par le système d'exploitation hôte. Sous Mac OS et Linux, il s'agit de la barre oblique (/). Sous Windows il s'agit de la barre oblique inverse (\).
`File.systemCharset`	Codage appliqué par défaut aux fichiers par le système d'exploitation hôte. Ce codage relève du jeu de caractères utilisé par le système d'exploitation et correspond à la langue en vigueur sur celui-ci.

La classe `Capabilities` comprend également des informations système qui peuvent être utiles lors de la manipulation des fichiers :

Propriété	Description
Capabilities.hasIME	Spécifie si le lecteur s'exécute sur un système qui dispose (`true`) ou non (`false`) d'un éditeur de méthode d'entrée (IME).
Capabilities.language	Indique le code de langue du système sur lequel s'exécute le lecteur.
Capabilities.os	Spécifie le système d'exploitation actuel.

Remarque : *faites preuve de prudence lorsque vous utilisez* `Capabilities.os` *pour déterminer les caractéristiques du système. S'il existe une propriété plus spécifique pour déterminer une caractéristique du système, utilisez-la. Vous risquez sinon d'écrire un code qui ne fonctionne pas correctement sur toutes les plates-formes. Considérons par exemple le code qui suit :*

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

Ce code cause des problèmes sous Linux. Il est préférable d'utiliser simplement la propriété `File.separator`.

Utilisation de répertoires

Adobe AIR 1.0 et les versions ultérieures

Le moteur d'exécution vous permet de manipuler les répertoires du système de fichiers local.

Pour plus de détails sur la création d'objets File qui pointent vers des répertoires, voir la section « Pointage d'un objet File vers un répertoire » à la page 696.

Création de répertoires

Adobe AIR 1.0 et les versions ultérieures

La méthode `File.createDirectory()` permet de créer un répertoire. Par exemple, le code suivant crée le répertoire AIR Test en tant que sous-répertoire du répertoire d'accueil de l'utilisateur :

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

Si le répertoire existe, la méthode `createDirectory()` n'agit pas.

Par ailleurs, dans certains modes, un objet FileStream crée des répertoires à l'ouverture des fichiers. Les répertoires qui n'existent pas sont créés lorsque vous instanciez une occurrence de FileStream en définissant le paramètre `fileMode` du constructeur `FileStream()` sur `FileMode.APPEND` ou `FileMode.WRITE`. Pour plus d'informations, voir la section « Flux de travail pour la lecture et l'écriture de fichiers » à la page 715.

Création d'un répertoire temporaire

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend la méthode `createTempDirectory()`, qui crée un répertoire dans le répertoire temporaire système, comme l'illustre l'exemple suivant :

```
var temp:File = File.createTempDirectory();
```

La méthode `createTempDirectory()` crée automatiquement un répertoire temporaire unique (ce qui vous évite d'avoir à déterminer un nouvel emplacement unique).

Vous pouvez utiliser un répertoire temporaire pour stocker provisoirement des fichiers temporaires utilisés au cours d'une session de l'application. Vous remarquerez que la méthode `createTempFile()` permet de créer des fichiers temporaires uniques dans le répertoire temporaire système.

Vous pouvez éventuellement supprimer le répertoire temporaire avant de fermer l'application, car cette opération n'est *pas* automatiquement effectuée sur tous les périphériques.

Enumération de répertoires

Adobe AIR 1.0 et les versions ultérieures

Les méthodes `getDirectoryListing()` ou `getDirectoryListingAsync()` d'un objet File permettent d'obtenir un tableau d'objets File pointant vers les fichiers et sous-répertoires d'un répertoire.

Par exemple, le code suivant répertorie le contenu du répertoire de documents de l'utilisateur (sans examiner les sous-répertoires) :

```
var directory:File = File.documentsDirectory;
var contents:Array = directory.getDirectoryListing();
for (var i:uint = 0; i < contents.length; i++)
{
    trace(contents[i].name, contents[i].size);
}
```

Lorsque vous utilisez la version asynchrone de la méthode, l'objet événement `directoryListing` possède la propriété `files`, qui consiste en un tableau d'objets File appartenant aux répertoires :

```
var directory:File = File.documentsDirectory;
directory.getDirectoryListingAsync();
directory.addEventListener(FileListEvent.DIRECTORY_LISTING, dirListHandler);

function dirListHandler(event:FileListEvent):void
{
    var contents:Array = event.files;
    for (var i:uint = 0; i < contents.length; i++)
    {
        trace(contents[i].name, contents[i].size);
    }
}
```

Copie et déplacement de répertoires

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez copier ou déplacer un répertoire, en utilisant les mêmes méthodes que pour un fichier. Par exemple, le code suivant copie un répertoire en mode synchrone :

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

Lorsque vous définissez le paramètre `overwrite` de la méthode `copyTo()` sur true, tous les fichiers et dossiers d'un répertoire cible existant sont supprimés et remplacés par les fichiers et dossiers du répertoire source (même si le fichier cible n'existe pas dans le répertoire source).

Le répertoire sur lequel vous définissez le paramètre `newLocation` de la méthode `copyTo()` spécifie le chemin d'accès au répertoire résultant, et *non* le répertoire *parent* qui contiendra le répertoire résultant.

Pour plus d'informations, voir la section « Copie et déplacement de fichiers » à la page 711.

Suppression du contenu d'un répertoire

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les méthodes `deleteDirectory()` et `deleteDirectoryAsync()`. Ces méthodes suppriment des répertoires. La première s'exécute en mode synchrone et la seconde en mode asynchrone (voir la section « Principes de base des classes File d'AIR » à la page 692). Elles comprennent toutes deux le paramètre `deleteDirectoryContents` (qui accepte une valeur booléenne). Lorsque ce paramètre est défini sur `true` (la valeur par défaut correspond à `false`), l'appel de la méthode supprime les répertoires non vides ; sinon, seuls les répertoires vides sont supprimés.

Par exemple, le code suivant supprime en mode synchrone le sous-répertoire AIR Test du répertoire de documents de l'utilisateur :

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

Le code suivant supprime en mode asynchrone le sous-répertoire AIR Test du répertoire de documents de l'utilisateur :

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Vous disposez également des méthodes `moveToTrash()` et `moveToTrashAsync()`, qui permettent de déplacer un répertoire vers la corbeille système. Pour plus d'informations, voir la section « Déplacement d'un fichier vers la corbeille » à la page 713.

Utilisation des fichiers

Adobe AIR 1.0 et les versions ultérieures

L'API de fichiers d'AIR vous permet d'ajouter des fonctionnalités de manipulation des fichiers de base à vos applications. Vous pouvez ainsi accéder à des fichiers en lecture ou en écriture, copier et supprimer des fichiers, etc. Comme vos applications ont accès au système de fichiers local, voir le chapitre « Sécurité AIR » à la page 1122, si ce n'est déjà fait.

Remarque : *vous pouvez associer un type de fichier à une application AIR (afin qu'un double-clic sur le fichier entraîne l'ouverture de l'application). Pour plus d'informations, voir la section « Gestion des associations de fichiers » à la page 920.*

Obtention d'informations sur les fichiers

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les propriétés suivantes qui fournissent des informations sur un fichier ou un répertoire vers lequel pointe un objet File :

Propriété File	Description
creationDate	Date de création du fichier sur le disque local.
creator	Obsolète. Utilisez la propriété `extension`. (Cette propriété renvoie le type de créateur Macintosh du fichier, qui est uniquement utilisé dans les versions de Mac OS antérieures à Mac OS X.)
downloaded	(AIR 2 et les versions ultérieures) Indique si le fichier ou le répertoire référencé a été téléchargé (depuis Internet) ou non. Cette propriété est utile uniquement sur les systèmes d'exploitation dans lesquels les fichiers peuvent être marqués comme téléchargés : • Windows XP Service Pack 2 et versions ultérieures, et Windows Vista • Mac OS 10.5 et versions ultérieures
exists	Indique si le fichier ou répertoire référencé existe.
extension	Extension de fichier, partie du nom qui suit (sans l'inclure) le point (« . »). Si le nom de fichier ne comprend pas de point, l'extension est `null`.
icon	Objet Icon contenant les icônes définies pour le fichier.
isDirectory	Indique si l'objet File référence un répertoire.
modificationDate	Date de la dernière modification du fichier ou du répertoire sur le disque local.
name	Nom du fichier ou répertoire (y compris l'éventuelle extension de fichier) sur le disque local.
nativePath	Chemin complet dans la représentation du système d'exploitation hôte (voir la section « Chemin des objets File » à la page 693).
parent	Dossier qui contient le dossier ou fichier représenté par l'objet File. Cette propriété est `null` si l'objet File référence un fichier ou un répertoire dans la racine du système de fichiers.
size	Taille du fichier sur le disque local, en octets.
type	Obsolète. Utilisez la propriété `extension`. (Sur le Macintosh, cette propriété correspond au type de fichier de quatre caractères qui est uniquement utilisé dans les versions de Mac OS antérieures à Mac OS X).
url	URL du fichier ou du répertoire (voir la section « Chemin des objets File » à la page 693).

Pour plus d'informations sur ces propriétés, voir l'entrée consacrée à la classe File dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Copie et déplacement de fichiers

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend deux méthodes permettant de copier des fichiers ou des répertoires : `copyTo()` et `copyToAsync()`. Elle propose également deux méthodes permettant de déplacer des fichiers ou des répertoires : `moveTo()` et `moveToAsync()`. Les méthodes `copyTo()` et `moveTo()` s'exécutent en mode synchrone, les méthodes `copyToAsync()` et `moveToAsync()` en mode asynchrone (voir la section « Principes de base des classes File d'AIR » à la page 692).

Pour copier ou déplacer un fichier, vous définissez deux objets File. L'un pointe vers le fichier à copier ou déplacer et appelle la méthode copy ou move. L'autre pointe vers le chemin de destination (résultant).

Le code suivant copie le fichier test.txt qui réside dans le sous-répertoire AIR Test du répertoire de documents de l'utilisateur vers le fichier copy.txt dans le même répertoire :

```
var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

Dans cet exemple, la valeur du deuxième paramètre, `overwrite`, de la méthode `copyTo()` est définie sur `true`. Si vous définissez le paramètre `overwrite` sur `true`, tout fichier cible existant est remplacé. Ce paramètre est facultatif. Si vous le définissez sur `false` (valeur par défaut), un événement IOErrorEvent est distribué si le fichier cible existe (et le fichier n'est pas copié).

La version « Async » des méthodes copy et move s'exécute en mode asynchrone. La méthode `addEventListener()` permet de surveiller la fin de la tâche ou les erreurs, comme l'illustre le code suivant :

```
var original = File.documentsDirectory;
original = original.resolvePath("AIR Test/test.txt");

var destination:File = File.documentsDirectory;
destination =  destination.resolvePath("AIR Test 2/copy.txt");

original.addEventListener(Event.COMPLETE, fileMoveCompleteHandler);
original.addEventListener(IOErrorEvent.IO_ERROR, fileMoveIOErrorEventHandler);
original.moveToAsync(destination);

function fileMoveCompleteHandler(event:Event):void {
    trace(event.target); // [object File]
}
function fileMoveIOErrorEventHandler(event:IOErrorEvent):void {
    trace("I/O Error.");
}
```

La classe File comprend également les méthodes `File.moveToTrash()` et `File.moveToTrashAsync()`, qui déplacent un fichier ou un répertoire vers la corbeille système.

Suppression d'un fichier

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les méthodes `deleteFile()` et `deleteFileAsync()`. Ces méthodes suppriment des fichiers. La première s'exécute en mode synchrone et la seconde en mode asynchrone (voir la section « Principes de base des classes File d'AIR » à la page 692).

Par exemple, le code suivant supprime, en mode synchrone, le fichier test.txt du répertoire de documents de l'utilisateur :

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

Le code suivant supprime, en mode asynchrone, le fichier test.txt du répertoire de documents de l'utilisateur :

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Vous disposez également des méthodes `moveToTrash()` et `moveToTrashAsync()`, qui permettent de déplacer un fichier ou un répertoire vers la corbeille système. Pour plus d'informations, voir la section « Déplacement d'un fichier vers la corbeille » à la page 713.

Déplacement d'un fichier vers la corbeille

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend les méthodes `moveToTrash()` et `moveToTrashAsync()`. Ces méthodes envoient un fichier ou un répertoire dans la corbeille système. La première s'exécute en mode synchrone et la seconde en mode asynchrone (voir la section « Principes de base des classes File d'AIR » à la page 692).

Par exemple, le code suivant déplace, en mode synchrone, le fichier test.txt du répertoire de documents de l'utilisateur vers la corbeille système :

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

Remarque : *sur les systèmes d'exploitation qui ne prennent pas en charge le concept de dossier de corbeille récupérable, ces fichiers sont immédiatement supprimés.*

Création d'un fichier temporaire

Adobe AIR 1.0 et les versions ultérieures

La classe File comprend la méthode `createTempFile()`, qui crée un fichier dans le répertoire temporaire système, comme l'illustre l'exemple suivant :

```
var temp:File = File.createTempFile();
```

La méthode `createTempFile()` crée automatiquement un fichier temporaire unique (ce qui vous évite d'avoir à déterminer un nouvel emplacement unique).

Vous pouvez utiliser un fichier temporaire pour stocker provisoirement des informations utilisées au cours d'une session de l'application. Vous remarquerez que vous disposez également de la méthode `createTempDirectory()`, qui permet de créer un répertoire temporaire unique dans le répertoire temporaire système.

Vous pouvez éventuellement supprimer le fichier temporaire avant de fermer l'application, car cette opération n'est *pas* automatiquement effectuée sur tous les périphériques.

Utilisation des volumes de stockage

Adobe AIR 2 et ultérieur

AIR 2 permet de détecter le montage ou le démontage des volumes de stockage de masse. La classe StorageVolumeInfo définit un objet `storageVolumeInfo` singleton. L'objet `StorageVolumeInfo.storageVolumeInfo` distribue un événement `storageVolumeMount` lors du montage d'un volume de stockage. Il distribue également un événement `storageVolumeUnmount` lors du démontage d'un volume. La classe StorageVolumeChangeEvent définit ces événements.

Remarque : *sur les distributions récentes de Linux, l'objet StorageVolumeInfo ne distribue les événements `storageVolumeMount` et `storageVolumeUnmount` que pour les périphériques physiques et les lecteurs réseau montés à des emplacements déterminés.*

La propriété `storageVolume` de la classe StorageVolumeChangeEvent est un objet StorageVolume. La classe StorageVolume définit les propriétés de base du volume de stockage :

- `drive` : lettre de lecteur d'un volume sous Windows (`null` sous les autres systèmes d'exploitation)
- `fileSystemType` : type du système de fichiers sur le volume de stockage (« FAT », « NTFS », « HFS » ou « UFS », par exemple).
- `isRemoveable` : indique si un volume peut être retiré (`true`) ou non (`false`).
- `isWritable` : indique si un volume est inscriptible (`true`) ou non (`false`).
- `name` : nom du volume
- `rootDirectory` : objet File correspondant au répertoire racine du volume

La classe StorageVolumeChangeEvent comprend également une propriété `rootDirectory`, à savoir un objet File qui référence le répertoire racine du volume de stockage monté ou démonté.

La propriété `storageVolume` de l'objet StorageVolumeChangeEvent n'est pas définie (autrement dit, elle correspond à `null`) si le volume est démonté. Vous pouvez toutefois accéder à la propriété `rootDirectory` de l'événement.

Le code suivant indique le nom et le chemin d'un volume de stockage lorsqu'il est monté :

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_MOUNT, onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

Le code suivant indique le chemin d'un volume de stockage lorsqu'il est démonté :

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_UNMOUNT, onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

L'objet `StorageVolumeInfo.storageVolumeInfo` comprend une méthode `getStorageVolumes()`. Cette méthode renvoie un vecteur d'objets StorageVolume qui correspondent aux volumes de stockage actuellement montés. Le code suivant indique comment afficher le nom et le répertoire racine de tous les volumes de stockage montés :

```
var volumes:Vector.<StorageVolume> = new Vector.<StorageVolume>;
volumes = StorageVolumeInfo.storageVolumeInfo.getStorageVolumes();
for (var i:int = 0; i < volumes.length; i++)
{
    trace(volumes[i].name, volumes[i].rootDirectory.nativePath);
}
```

Remarque : *sur les distributions récentes de Linux, la méthode `getStorageVolumes()` renvoie des objets correspondant à des périphériques physiques et à des lecteurs réseau montés à des emplacements déterminés.*

La méthode `File.getRootDirectories()` recense les répertoires racine (voir « Pointage vers la racine du système de fichiers » à la page 698). Les objets StorageVolume (énumérés par la méthode `StorageVolumeInfo.getStorageVolumes()`) proposent toutefois plus d'informations sur les volumes de stockage.

Vous disposez de la propriété `spaceAvailable` de la propriété `rootDirectory` d'un objet StorageVolume pour obtenir l'espace disponible sur un volume de stockage. (Voir « Détermination de l'espace disponible sur un volume » à la page 706.)

Voir aussi

StorageVolume

StorageVolumeInfo

Lecture et écriture de fichiers

Adobe AIR 1.0 et les versions ultérieures

La classe FileStream permet aux applications AIR de lire et d'écrire dans le système de fichiers.

Flux de travail pour la lecture et l'écriture de fichiers

Adobe AIR 1.0 et les versions ultérieures

Le flux de travail de lecture et d'écriture de fichier est décrit ci-après.

Initialisez un objet File pointant vers le chemin.
Cet objet File représente le chemin du fichier que vous souhaitez utiliser (ou d'un fichier que vous créerez ultérieurement).

```
var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

Cet exemple utilise la propriété `File.documentsDirectory` et la méthode `resolvePath()` d'un objet File pour initialiser celui-ci. Toutefois, vous pouvez pointer un objet File vers un fichier de plusieurs autres façons. Pour plus d'informations, voir la section « Pointage d'un objet File vers un fichier » à la page 700.

Initialisez un objet FileStream.

Appelez la méthode open() ou openAsync() de l'objet FileStream.
La méthode que vous appelez varie selon que vous souhaitez ouvrir le fichier en mode synchrone ou asynchrone. Utilisez l'objet File comme paramètre `file` de la méthode open. Pour le paramètre `fileMode`, spécifiez une constante de la classe FileMode qui définit le mode d'utilisation du fichier.

Par exemple, le code suivant initialise un objet FileStream utilisé pour créer un fichier et, éventuellement, remplacer les données existantes :

```
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

Pour plus d'informations, voir les sections « Initialisation d'un objet FileStream et ouverture et fermeture de fichiers » à la page 717 et « Modes d'ouverture FileStream » à la page 716.

Si vous avez ouvert le fichier en mode asynchrone (à l'aide de la méthode openAsync()), ajoutez et définissez des écouteurs d'événement pour l'objet FileStream.
Ces méthodes d'écouteurs d'événements réagissent aux événements distribués par l'objet FileStream dans divers cas de figure. Parmi ces cas de figure figurent notamment la lecture de données dans le fichier, la génération d'erreurs E/S ou l'écriture de l'intégralité des données.

Pour plus d'informations, voir la section « Programmation asynchrone et événements générés par un objet FileStream ouvert en mode asynchrone » à la page 721.

Incluez du code permettant de lire et d'écrire des données, le cas échéant.
La classe FileStream propose de nombreuses méthodes relatives à la lecture ou l'écriture (elles commencent toutes par « read » ou « write »). La méthode que vous choisissez pour lire et écrire des données varie en fonction du format des données dans le fichier cible.

Par exemple, si les données du fichier cible sont au format texte UTF, vous pouvez utiliser les méthodes `readUTFBytes()` et `writeUTFBytes()`. Pour manipuler les données en tant que tableaux d'octets, utilisez les méthodes `readByte()`, `readBytes()`, `writeByte()` et `writeBytes()`. Pour plus d'informations, voir la section « Format de données et choix des méthodes de lecture et d'écriture » à la page 722.

Si vous avez ouvert le fichier en mode asynchrone, veillez à ce qu'un volume suffisant de données soit disponible avant d'appeler une méthode de lecture. Pour plus d'informations, voir la section « Mémoire tampon de lecture et propriété bytesAvailable d'un objet FileStream » à la page 719.

Si, avant d'accéder en écriture à un fichier, vous souhaitez vérifier la quantité d'espace disque disponible, vous pouvez consulter la propriété spaceAvailable de l'objet File. Pour plus d'informations, voir la section « Détermination de l'espace disponible sur un volume » à la page 706.

Appelez la méthode close() de l'objet FileStream lorsque vous avez terminé de manipuler le fichier.
D'autres applications peuvent alors accéder à ce dernier.

Pour plus d'informations, voir la section « Initialisation d'un objet FileStream et ouverture et fermeture de fichiers » à la page 717.

Vous trouverez un exemple d'application utilisant la classe FileStream pour accéder en lecture et en écriture à des fichiers dans les articles suivants du Centre des développeurs Adobe AIR :

- Développement d'un éditeur de fichier texte
- Développement d'un éditeur de fichier texte
- Lecture et écriture à partir d'un fichier de préférences XML

Utilisation des objets FileStream

Adobe AIR 1.0 et les versions ultérieures

La classe FileStream définit des méthodes permettant d'ouvrir des fichiers et d'y accéder en lecture ou en écriture.

Modes d'ouverture FileStream

Adobe AIR 1.0 et les versions ultérieures

Les méthodes `open()` et `openAsync()` d'un objet FileStream comprennent le paramètre `fileMode`, qui définit certaines propriétés d'un flux de fichier, pour indiquer notamment :

- s'il est possible d'accéder au fichier en lecture ;
- s'il est possible d'accéder au fichier en écriture ;
- si les données sont systématiquement ajoutées à la fin du fichier (lors de l'écriture) ;
- que faire lorsque le fichier n'existe pas (et que ses répertoires parent n'existent pas).

Les différents modes de fichier disponibles (que vous pouvez spécifier dans le paramètre `fileMode` des méthodes `open()` et `openAsync()`) sont les suivants :

Mode de fichier	Description
FileMode.READ	Indique que le fichier est ouvert à des fins de lecture seulement.
FileMode.WRITE	Indique que le fichier est ouvert à des fins d'écriture. Si le fichier n'existe pas, il est créé à l'ouverture de l'objet FileStream. S'il existe, les données existantes sont supprimées.
FileMode.APPEND	Indique que le fichier est ouvert et qu'il est possible d'ajouter des données en fin de fichier. Si le fichier n'existe pas, il est créé. S'il existe, les données existantes ne sont pas remplacées. L'écriture commence à la fin du fichier.
FileMode.UPDATE	Indique que le fichier est ouvert à des fins de lecture et d'écriture. Si le fichier n'existe pas, il est créé. Choisissez ce mode pour un accès en lecture/écriture aléatoire au fichier. Vous pouvez lire à partir de tout emplacement du fichier. Lorsque vous écrivez dans celui-ci, seuls les octets écrits remplacent les octets existants (aucun autre octet n'étant affecté).

Initialisation d'un objet FileStream et ouverture et fermeture de fichiers

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous ouvrez un objet FileStream, vous autorisez un accès en lecture ou en écriture à un fichier. Pour ouvrir un objet FileStream, vous transmettez un objet File à sa méthode `open()` ou `openAsync()` :

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

Le paramètre `fileMode` (le deuxième des méthodes `open()` et `openAsync()`) détermine si le fichier est ouvert à des fins de lecture, écriture, ajout en fin de fichier ou mise à jour. Pour plus d'informations, voir la section précédente, « Modes d'ouverture FileStream » à la page 716.

Si vous ouvrez le fichier en mode asynchrone à l'aide de la méthode `openAsync()`, configurez des écouteurs d'événement pour gérer les événements asynchrones :

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completeHandler);
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.addEventListener(IOErrorEvent.IO_Error, errorHandler);
myFileStream.openAsync(myFile, FileMode.READ);

function completeHandler(event:Event):void {
    // ...
}

function progressHandler(event:ProgressEvent):void {
    // ...
}

function errorHandler(event:IOErrorEvent):void {
    // ...
}
```

Le fichier est ouvert en mode synchrone ou asynchrone, selon que vous utilisez la méthode `open()` ou `openAsync()`. Pour plus d'informations, voir la section « Principes de base des classes File d'AIR » à la page 692.

Si vous définissez le paramètre `fileMode` sur `FileMode.READ` ou `FileMode.UPDATE` dans la méthode d'ouverture de l'objet FileStream, les données sont lues et insérées dans la mémoire tampon de lecture dès l'ouverture de l'objet FileStream. Pour plus d'informations, voir la section « Mémoire tampon de lecture et propriété bytesAvailable d'un objet FileStream » à la page 719.

Vous pouvez appeler la méthode `close()` d'un objet FileStream pour fermer le fichier associé, ce qui le met à la disposition d'autres applications.

Propriété position d'un objet FileStream

Adobe AIR 1.0 et les versions ultérieures

La propriété `position` d'un objet FileStream détermine l'emplacement de lecture ou d'écriture des données de la méthode de lecture ou d'écriture suivante.

Préalablement à une opération de lecture ou d'écriture, définissez la propriété `position` sur une position valide dans le fichier.

Par exemple, le code suivant écrit la chaîne `"hello"` (au codage UTF) à la position 8 dans le fichier :

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.UPDATE);
myFileStream.position = 8;
myFileStream.writeUTFBytes("hello");
```

Lorsque vous ouvrez initialement un objet FileStream, la propriété `position` est définie sur 0.

Avant une opération de lecture, la valeur de la propriété `position` doit être comprise entre 0 et le nombre maximal d'octets du fichier (autrement dit, une position existante dans le fichier).

La valeur de la propriété `position` est uniquement modifiée dans les cas suivants :

- Vous définissez explicitement la propriété `position`.
- Vous appelez une méthode de lecture.
- Vous appelez une méthode d'écriture.

Lorsque vous appelez une méthode de lecture ou d'écriture d'un objet FileStream, la propriété `position` est immédiatement incrémentée du nombre d'octets lus ou écrits. Selon la méthode de lecture utilisée, la propriété `position` est incrémentée du nombre d'octets que vous spécifiez pour la lecture ou du nombre d'octets disponibles. Lorsque, par la suite, vous appelez une méthode de lecture ou d'écriture, elle commence la lecture ou l'écriture à la nouvelle position.

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.UPDATE);
myFileStream.position = 4000;
trace(myFileStream.position); // 4000
myFileStream.writeBytes(myByteArray, 0, 200);
trace(myFileStream.position); // 4200
```

Il existe cependant une exception : si un objet FileStream est ouvert en mode d'ajout en fin de fichier, la propriété `position` ne change pas à la suite de l'appel d'une méthode d'écriture. (Dans ce mode, les données sont toujours ajoutées à la fin du fichier, quelle que soit la valeur de la propriété `position`.)

Si un fichier est ouvert en mode asynchrone, l'opération d'écriture ne se termine pas avant l'exécution de la ligne de code suivante. Vous pouvez néanmoins appeler successivement plusieurs méthodes asynchrones : le moteur d'exécution les exécute dans l'ordre.

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.openAsync(myFile, FileMode.WRITE);
myFileStream.writeUTFBytes("hello");
myFileStream.writeUTFBytes("world");
myFileStream.addEventListener(Event.CLOSE, closeHandler);
myFileStream.close();
trace("started.");

closeHandler(event:Event):void
{
    trace("finished.");
}
```

La sortie de suivi de ce code est la suivante :

```
started.
finished.
```

Il est *possible* de spécifier la valeur de la propriété `position` immédiatement après l'appel d'une méthode de lecture ou d'écriture (ou à tout moment). L'opération de lecture ou d'écriture suivante commence alors à cette position. Par exemple, vous remarquerez que le code suivant définit la propriété `position` immédiatement après un appel de l'opération `writeBytes()` et que la `position` est définie sur cette valeur (300) même à l'issue de l'opération d'écriture :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.openAsync(myFile, FileMode.UPDATE);
myFileStream.position = 4000;
trace(myFileStream.position); // 4000
myFileStream.writeBytes(myByteArray, 0, 200);
myFileStream.position = 300;
trace(myFileStream.position); // 300
```

Mémoire tampon de lecture et propriété bytesAvailable d'un objet FileStream

Adobe AIR 1.0 et les versions ultérieures

Lors de l'ouverture d'un objet FileStream doté de fonctionnalités de lecture (le paramètre `fileMode` de sa méthode `open()` ou `openAsync()` étant défini sur `READ` ou `UPDATE`), le moteur d'exécution stocke les données dans une mémoire tampon interne. L'objet FileStream commence la lecture et l'insertion des données dans la mémoire tampon dès l'ouverture du fichier (par appel de la méthode `open()` ou `openAsync()` de l'objet FileStream).

Si un fichier est ouvert en mode synchrone (à l'aide de la méthode `open()`), vous pouvez définir le pointeur `position` sur n'importe quelle position valide (dans les limites du fichier) et commencer à lire tout volume de données (dans les limites du fichier), comme l'illustre le code suivant (qui considère comme acquis que le fichier contient au moins 100 octets) :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
myFileStream.position = 10;
myFileStream.readBytes(myByteArray, 0, 20);
myFileStream.position = 89;
myFileStream.readBytes(myByteArray, 0, 10);
```

Qu'un fichier soit ouvert en mode synchrone ou asynchrone, les méthodes de lecture lisent systématiquement les octets « disponibles », représentés par la propriété `bytesAvalable`. En mode de lecture synchrone, tous les octets du fichier sont disponibles en permanence. En mode asynchrone, les octets deviennent disponibles à partir de la position indiquée par la propriété `position`, dans une série de remplissages asynchrones de la mémoire tampon signalés par des événements `progress`.

En mode *synchrone*, la propriété `bytesAvailable` représente systématiquement le nombre d'octets compris entre la propriété `position` et la fin du fichier (tous les octets du fichier sont toujours disponibles pour la lecture).

En mode *asynchrone*, vous devez vous assurer que la mémoire tampon de lecture contient suffisamment de données avant d'appeler une méthode de lecture. Dans ce mode, au fur et à mesure de la lecture, les données du fichier sont ajoutées à la mémoire tampon, à partir de la `position` spécifiée au début de l'opération de lecture, et la propriété `bytesAvailable` est incrémentée à chaque octet lu. La propriété `bytesAvailable` indique le nombre d'octets disponibles entre l'octet situé à la position spécifiée par la propriété `position` et la fin de la mémoire tampon. L'objet FileStream envoie régulièrement un événement `progress`.

En mode asynchrone, l'objet FileStream distribue régulièrement l'événement `progress` à mesure que des données sont disponibles dans la mémoire tampon de lecture. Par exemple, le code suivant insère des données dans un objet ByteArray, `bytes`, au fur et à mesure de leur insertion dans la mémoire tampon de lecture :

```
 var bytes:ByteArray = new ByteArray();
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.openAsync(myFile, FileMode.READ);

function progressHandler(event:ProgressEvent):void
{
    myFileStream.readBytes(bytes, myFileStream.position, myFileStream.bytesAvailable);
}
```

En mode asynchrone, seules les données que contient la mémoire tampon de lecture peuvent être lues. En outre, les données sont supprimées de la mémoire tampon au fur et à mesure de leur lecture. Avant d'appeler une opération de lecture, vous devez donc vous assurer que les données existent dans la mémoire tampon de lecture. Par exemple, le code suivant lit 8 000 octets de données à partir de la position 4 000 dans le fichier :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
myFileStream.position = 4000;

var str:String = "";

function progressHandler(event:Event):void
{
    if (myFileStream.bytesAvailable > 8000 )
    {
        str += myFileStream.readMultiByte(8000, "iso-8859-1");
    }
}
```

Lors d'une opération d'écriture, l'objet FileStream n'insère pas de données dans la mémoire tampon de lecture. Au terme de l'opération (toutes les données de la mémoire tampon d'écriture étant écrites dans le fichier), l'objet FileStream commence une nouvelle mémoire tampon de lecture (en supposant que l'objet FileStream ouvert associé possède des fonctionnalités de lecture) et commence à insérer des données dans la mémoire tampon de lecture à partir de la position spécifiée par la propriété `position`. La propriété `position` peut correspondre à la position du dernier octet écrit, mais elle est différente si l'utilisateur lui affecte une autre valeur après l'opération d'écriture.

Programmation asynchrone et événements générés par un objet FileStream ouvert en mode asynchrone

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un fichier est ouvert en mode asynchrone (à l'aide de la méthode `openAsync()`), la lecture et l'écriture de fichiers sont asynchrones. Un autre code ActionScript peut s'exécuter au fur et à mesure de l'insertion de données dans la mémoire tampon de lecture et de l'écriture de données de sortie.

Vous devez donc vous enregistrer pour les événements générés par l'objet FileStream ouvert en mode asynchrone.

En vous enregistrant pour l'événement `progress`, vous pouvez être averti dès que de nouvelles données sont disponibles à des fins de lecture, comme l'illustre le code suivant :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(ProgressEvent.PROGRESS, progressHandler);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";

function progressHandler(event:ProgressEvent):void
{
    str += myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

Vous pouvez lire la totalité des données en vous enregistrant pour l'événement `complete`, comme l'illustre le code suivant :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";
function completeHandler(event:Event):void
{
    str = myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

De même que des données en entrée sont mises en mémoire tampon à des fins de lecture asynchrone, les données que vous écrivez dans un flux asynchrone sont mises en mémoire tampon et écrites dans le fichier de manière asynchrone. Alors que des données sont écrites dans un fichier, l'objet FileStream distribue régulièrement un objet `OutputProgressEvent`. Un objet `OutputProgressEvent` comprend une propriété `bytesPending` définie sur le nombre d'octets restants à écrire. Vous pouvez vous enregistrer pour l'événement `outputProgress` afin d'être averti lorsque cette mémoire tampon est écrite dans le fichier, peut-être pour afficher une boîte de dialogue de progression. Toutefois, cette opération n'est généralement pas nécessaire. Vous pouvez notamment appeler la méthode `close()` sans vous soucier des octets non écrits. L'objet FileStream continue d'écrire des données et l'événement `close` est envoyé une fois le dernier octet écrit dans le fichier et le fichier sous-jacent fermé.

Format de données et choix des méthodes de lecture et d'écriture

Adobe AIR 1.0 et les versions ultérieures

Tout fichier est un ensemble d'octets sur disque. Dans ActionScript, les données d'un fichier peuvent toujours être représentées sous la forme d'un objet ByteArray. Par exemple, le code suivant lit les données d'un fichier et les insère dans un objet ByteArray nommé `bytes` :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completeHandler);
myFileStream.openAsync(myFile, FileMode.READ);
var bytes:ByteArray = new ByteArray();

function completeHandler(event:Event):void
{
    myFileStream.readBytes(bytes, 0, myFileStream.bytesAvailable);
}
```

De même, le code suivant écrit les données de l'objet ByteArray `bytes` dans un fichier :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.WRITE);
myFileStream.writeBytes(bytes, 0, bytes.length);
```

Cependant, vous souhaitez rarement stocker les données dans un objet ByteArray ActionScript. Les fichiers de données sont souvent dans un format de fichier spécifique.

Il se peut par exemple que les données du fichier soient au format texte et que vous souhaitiez les représenter dans un objet String.

C'est pourquoi la classe FileStream comprend des méthodes de lecture et d'écriture de données de types autres que les objets ByteArray. Par exemple, la méthode `readMultiByte()` permet de lire des données dans un fichier et de les stocker dans une chaîne, comme l'illustre le code suivant :

```
 var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.addEventListener(Event.COMPLETE, completed);
myFileStream.openAsync(myFile, FileMode.READ);
var str:String = "";

function completeHandler(event:Event):void
{
    str = myFileStream.readMultiByte(myFileStream.bytesAvailable, "iso-8859-1");
}
```

Le deuxième paramètre de la méthode `readMultiByte()` spécifie le format texte utilisé par ActionScript pour interpréter les données (« iso-8859-1 », en l'occurrence). Adobe AIR prend en charge les encodages de jeux de caractères standard (voir Jeux de caractères pris en charge).

La classe FileStream comprend également la méthode `readUTFBytes()`, qui lit des données de la mémoire tampon de lecture et les insère dans une chaîne à l'aide du jeu de caractères UTF-8. Les caractères du jeu de caractères UTF-8 sont de longueur variable. N'utilisez donc pas `readUTFBytes()` dans une méthode répondant à l'événement `progress` car les données situées à la fin de la mémoire tampon de lecture sont susceptibles de représenter un caractère incomplet. (Cette règle s'applique également à l'utilisation de la méthode `readMultiByte()` avec un codage de caractères de longueur variable.) Lorsque l'objet FileStream distribue l'événement `progress`, vous devez donc lire l'ensemble complet de données.

Vous disposez également de méthodes d'écriture similaires, `writeMultiByte()` et `writeUTFBytes()`, pour manipuler les objets String et les fichiers de texte.

Les méthodes `readUTF()` et `writeUTF()` (à ne pas confondre avec `readUTFBytes()` et `writeUTFBytes()`) permettent également la lecture ou l'écriture de données dans un fichier, mais elles considèrent comme acquis que des données indiquant la longueur des données texte précèdent ces dernières, ce qui n'est pas une pratique courante dans les fichiers de texte standard.

Certains fichiers de texte UTF commencent par un caractère « UTF-BOM » (de l'anglais Byte Order Mark) qui définit le boutisme, ainsi que le format de codage (UTF-16 ou UTF-32).

Vous trouverez un exemple de lecture et d'écriture dans un fichier de texte à la section « Exemple : Lecture et insertion d'un fichier XML dans un objet XML » à la page 725.

Les méthodes `readObject()` et `writeObject()` facilitent le stockage et la récupération de données pour les objets ActionScript complexes. Les données sont codées au format AMF (ActionScript Message Format). Adobe AIR, Flash Player, Flash Media Server et Flex Data Services comprennent des API permettant de manipuler les données dans ce format.

Il existe d'autres méthodes de lecture et d'écriture, notamment `readDouble()` et `writeDouble()`. Cependant si vous les utilisez, assurez-vous que le format de fichier correspond au format des données définies par ces méthodes.

Les formats de fichier sont souvent plus complexes que de simples formats de texte. Par exemple, un fichier MP3 contient des données compressées qui peuvent uniquement être interprétées par le biais des algorithmes de décompression et de décodage spécifiques à ce format. Les fichiers MP3 peuvent également comprendre des balises ID3 qui contiennent des informations de métabalise concernant le fichier (tel que le titre et l'interprète d'une chanson). Il existe plusieurs versions du format ID3, mais le plus simple (ID3 version 1) est présenté à la section « Exemple : Lecture et écriture de données en mode aléatoire » à la page 726.

D'autres formats de fichier (pour les images, bases de données, documents d'application, etc.) ont une structure de données différente. Vous devez comprendre celle-ci pour pouvoir les utiliser dans ActionScript.

Utilisation des méthodes load() et save()

Flash Player 10 et les versions ultérieures, Adobe AIR 1.5 et les versions ultérieures

Flash Player 10 a ajouté les méthodes `load()` et `save()` dans la classe FileReference. Ces méthodes sont également présentes dans AIR 1.5, et la classe File hérite des méthodes de la classe FileReference. L'objectif de ces méthodes est de fournir aux utilisateurs un moyen sécurisé pour charger et enregistrer les données des fichiers dans Flash Player. Toutefois, les applications AIR peuvent également les utiliser pour charger et enregistrer les fichiers de façon asynchrone.

Par exemple, le code suivant enregistre une chaîne dans un fichier texte :

```
var file:File = File.applicationStorageDirectory.resolvePath("test.txt");
var str:String = "Hello.";
file.addEventListener(Event.COMPLETE, fileSaved);
file.save(str);
function fileSaved(event:Event):void
{
    trace("Done.");
}
```

Le paramètre `data` de la méthode `save()` peut prendre une valeur String, XML ou ByteArray. Lorsque l'argument est une valeur String ou XML, la méthode enregistre le fichier sous forme de fichier texte au format UTF-8.

Lorsque cet exemple de code s'exécute, l'application présente une boîte de dialogue qui permet à l'utilisateur de sélectionner la destination du fichier enregistré.

Le code suivant charge une chaîne à partir d'un fichier texte UTF-8 :

```
var file:File = File.applicationStorageDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, loaded);
file.load();
var str:String;
function loaded(event:Event):void
{
    var bytes:ByteArray = file.data;
    str = bytes.readUTFBytes(bytes.length);
    trace(str);
}
```

La classe FileStream offre plus de fonctionnalités que celles fournies par les méthodes `load()` et `save()` :

- La classe FileStream vous permet de lire et d'écrire des données de façon synchrone et asynchrone.
- La classe FileStream vous permet d'écrire dans un fichier de façon incrémentielle.
- La classe FileStream vous permet d'ouvrir un fichier en mode aléatoire (lecture et écriture dans toute section du fichier).
- La classe FileStream vous permet de spécifier le type d'accès au fichier disponible pour le fichier via la définition du paramètre `fileMode` de la méthode `open()` ou `openAsync()`.
- La classe FileStream vous permet d'enregistrer des données dans des fichiers sans présenter à l'utilisateur une boîte de dialogue Ouvrir ou Enregistrer.
- Lors de la lecture de données avec la classe FileStream, vous pouvez utiliser directement des types autres que des tableaux d'octets.

Exemple : Lecture et insertion d'un fichier XML dans un objet XML

Adobe AIR 1.0 et les versions ultérieures

Les exemples suivants expliquent comment accéder en lecture et en écriture à un fichier de texte contenant des données XML.

Pour lire le fichier, initialisez les objets File et FileStream, appelez la méthode `readUTFBytes()` de l'objet FileStream et convertissez la chaîne en objet XML :

```
var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.READ);
var prefsXML:XML = XML(fileStream.readUTFBytes(fileStream.bytesAvailable));
fileStream.close();
```

De même, pour écrire dans le fichier, il suffit de définir les objets File et FileStream appropriés, puis d'appeler une méthode d'écriture de l'objet FileStream. Transmettez la version chaîne des données XML à la méthode d'écriture, comme l'illustre le code suivant :

```
var prefsXML:XML = <prefs><autoSave>true</autoSave></prefs>;
var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
fileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);

var outputString:String = '<?xml version="1.0" encoding="utf-8"?>\n';
outputString += prefsXML.toXMLString();

fileStream.writeUTFBytes(outputString);
fileStream.close();
```

Ces exemples utilisent les méthodes `readUTFBytes()` et `writeUTFBytes()`, car il est considéré comme acquis que les fichiers sont au format UTF-8. Si tel n'est pas le cas, vous devrez peut-être recourir à une autre méthode (voir la section « Format de données et choix des méthodes de lecture et d'écriture » à la page 722).

Les exemples précédents utilisent des objets FileStream ouverts en mode synchrone. Vous pouvez aussi ouvrir des fichiers en mode asynchrone (ce type d'opération nécessite l'utilisation de fonctions d'écouteur d'événement pour répondre aux événements). Par exemple, le code suivant illustre comment lire un fichier XML en mode asynchrone :

```
var file:File = File.documentsDirectory.resolvePath("AIR Test/preferences.xml");
var fileStream:FileStream = new FileStream();
fileStream.addEventListener(Event.COMPLETE, processXMLData);
fileStream.openAsync(file, FileMode.READ);
var prefsXML:XML;

function processXMLData(event:Event):void
{
    prefsXML = XML(fileStream.readUTFBytes(fileStream.bytesAvailable));
    fileStream.close();
}
```

La méthode `processXMLData()` est appelée lorsque le fichier est intégralement lu et inséré dans la mémoire tampon de lecture (autrement dit, lorsque l'objet FileStream distribue l'événement `complete`). Elle appelle la méthode `readUTFBytes()` pour obtenir une version chaîne des données lues, puis elle crée un objet XML, `prefsXML`, basé sur cette chaîne.

Pour accéder à un exemple d'application possédant ces fonctionnalités, voir Lecture et écriture à partir d'un fichier de préférence XML.

Exemple : Lecture et écriture de données en mode aléatoire

Adobe AIR 1.0 et les versions ultérieures

Les fichiers MP3 peuvent renfermer des balises ID3, autrement dit, des sections en début ou fin de fichier contenant des métadonnées identifiant l'enregistrement. Il existe différentes versions du format de balise ID3. L'exemple suivant explique comment accéder en lecture et écriture à un fichier MP3 contenant le format ID3 le plus simple (ID3 version 1.0) par le biais d'un « accès aléatoire aux données », c'est-à-dire en lisant et écrivant des données à des emplacements arbitraires dans le fichier.

Dans un fichier MP3 contenant une balise ID3 version 1, les données ID3 figurent dans les 128 derniers octets du fichier.

Lorsque vous accédez à un fichier en mode de lecture/écriture aléatoire, il est important de définir le paramètre `fileMode` de la méthode `open()` ou`openAsync()` sur `FileMode.UPDATE` :

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

Vous pouvez ainsi accéder au fichier en lecture et écriture.

A l'ouverture du fichier, vous pouvez définir le pointeur `position` sur 128 octets avant la fin du fichier :

```
fileStr.position = file.size - 128;
```

Ce code définit la propriété `position` sur cet emplacement du fichier car le format ID3 v1.0 stipule que les balises ID3 sont stockées dans les derniers 128 octets du fichier. La spécification définit également les informations suivantes :

- Les 3 premiers octets de la balise représentent la chaîne `"TAG"`.
- Les 30 caractères suivants représentent le titre de la piste MP3, sous forme de chaîne.
- Les 30 caractères suivants représentent le nom de l'interprète, sous forme de chaîne.
- Les 30 caractères suivants représentent le nom de l'album, sous forme de chaîne.
- Les 4 caractères suivants représentent l'année, sous forme de chaîne.
- Les 30 caractères suivants représentent le commentaire, sous forme de chaîne.
- L'octet suivant contient un code identifiant le genre de la piste.
- Toutes les données texte sont au format ISO 8859-1.

La méthode `id3TagRead()` vérifie les données après leur lecture (sur envoi de l'événement `complete`) :

```
function id3TagRead():void
{
    if (fileStr.readMultiByte(3, "iso-8859-1").match(/tag/i))
    {
        var id3Title:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Artist:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Album:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3Year:String = fileStr.readMultiByte(4, "iso-8859-1");
        var id3Comment:String = fileStr.readMultiByte(30, "iso-8859-1");
        var id3GenreCode:String =  fileStr.readByte().toString(10);
    }
}
```

Vous pouvez également écrire dans le fichier en mode aléatoire. Par exemple, vous pourriez analyser la variable `id3Title` pour vérifier que l'utilisation des majuscules est correcte (à l'aide des méthodes de la classe String), puis écrire une chaîne modifiée, `newTitle`, dans le fichier, comme l'illustre l'exemple suivant :

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

Conformément à la norme ID3 version 1, la chaîne `newTitle` doit comprendre 30 caractères et être remplie à droite par le code de caractère 0 (`String.fromCharCode(0)`).

Chapitre 39 : Stockage des données locales

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe SharedObject permet de stocker des volumes réduits de données sur l'ordinateur client. Dans Adobe AIR, vous disposez également de la classe EncryptedLocalStore pour stocker des volumes réduits de données utilisateur confidentielles sur l'ordinateur local dans une application AIR.

Il est également possible de lire et d'écrire des fichiers dans le système de fichiers et, dans Adobe AIR, d'accéder aux fichiers de base de données locaux. Pour plus d'informations, voir « Utilisation du système de fichiers » à la page 676 et « Utilisation des bases de données SQL locales dans AIR » à la page 741.

Diverses considérations liées à la sécurité affectent les objets partagés. Pour plus d'informations, voir « Objets partagés » à la page 1120 dans « Sécurité » à la page 1085.

Objets partagés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un objet partagé, parfois appelé cookie Flash, est un fichier de données qui peut être créé sur votre ordinateur par les sites que vous visitez. Les objets partagés servent le plus souvent à optimiser la navigation sur le Web, par exemple en vous permettant de personnaliser l'aspect d'un site Web que vous consultez fréquemment.

A propos des objets partagés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les objets partagés fonctionnent comme des cookies de navigation. La classe SharedObject permet de stocker des données sur le disque dur local de l'utilisateur et d'appeler ces données pendant la même session ou dans le cadre d'une session ultérieure. Les applications peuvent accéder uniquement à leurs propres données SharedObject, et ce, uniquement si elles sont exécutées sur le même domaine. Les données ne sont pas transmises au serveur et les applications exécutées dans d'autres domaines ne peuvent pas y accéder. En revanche, les applications du même domaine sont en mesure d'y accéder.

Comparaison des objets partagés avec les cookies

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les cookies et les objets partagés sont très similaires. Comme la plupart des programmateurs Web connaissent le fonctionnement des cookies, la comparaison des cookies et des objets SharedObjects locaux peut s'avérer utile.

Les cookies conformes au standard RFC 2109 possèdent habituellement les propriétés suivantes :

- Ils possèdent une date d'expiration, souvent fixée par défaut à la fin d'une session.
- Ils sont désactivables par le client en fonction de sites spécifiques.
- La limite totale de cookies est de 300, avec 20 cookies maximum par site.
- Leur taille est habituellement limitée à 4 Ko chacun.

- Ils sont parfois considérés comme une menace à la sécurité et sont, par conséquent, parfois désactivés sur le client.
- Ils sont stockés dans un emplacement spécifié par le navigateur client.
- Ils sont transmis du client au serveur via HTTP.

  Par contre, les objets partagés possèdent les propriétés suivantes :

- Ils n'expirent pas par défaut.
- Par défaut, leur taille est limitée à 100 Ko chacun.
- Ils peuvent enregistrer des types de données simples (comme String, Array et Date).
- Ils sont stockés à un emplacement spécifié par l'application (dans le répertoire de base de l'utilisateur).
- Ils ne sont jamais transmis entre le client et le serveur.

A propos de la classe SharedObject

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe SharedObject permet de créer et de supprimer des objets partagés, ainsi que de détecter la taille actuelle d'un objet SharedObject en cours d'utilisation.

Création d'un objet partagé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour créer un objet SharedObject, faites appel à la méthode `SharedObject.getLocal()`, dont la syntaxe est la suivante :

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

L'exemple suivant crée un objet partagé dénommé mySO :

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

Cette opération crée un fichier sur l'ordinateur du client dénommé preferences.sol.

Le terme *local* fait référence à l'emplacement de l'objet partagé. Dans ce cas, Adobe® Flash® Player enregistre le fichier SharedObject localement dans le répertoire de base du client.

A la création d'un objet partagé, Flash Player crée un nouveau répertoire pour l'application et le domaine dans son sandbox. Il crée également un fichier *.sol qui enregistre les données SharedObject. L'emplacement par défaut de ce fichier est un sous-répertoire du répertoire de base de l'utilisateur. Le tableau suivant indique les emplacements par défaut de ce répertoire :

Système d'exploitation	Emplacement
Windows 95/98/ME/2000/XP	`c:/Documents and Settings/`*`username`*`/Application Data/Macromedia/Flash Player/#SharedObjects`
Windows Vista/Windows 7	`c:/Users/`*`username`*`/AppData/Roaming/Macromedia/Flash Player/#SharedObjects`
Macintosh OS X	`/Users/`*`username`*`/Library/Preferences/Macromedia/Flash Player/#SharedObjects/`*`web_domain/path_to_application/application_name/object_name`*`.sol`
Linux/Unix	`/home/username/.macromedia/Flash_Player/#SharedObjects/`*`web_domain/path_to_application/application_name/object_name`*`.sol`

Sous le répertoire #SharedObjects se trouve un répertoire nommé de façon aléatoire. Sous ce dernier se trouve un répertoire correspondant au nom d'hôte, puis le chemin vers l'application et finalement le fichier *.sol.

Par exemple, si vous demandez un application nommée MyApp.swf sur l'hôte local et dans un sous-répertoire nommé /sos, Flash Player enregistre le fichier *.sol à l'emplacement suivant sur Windows XP:

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

Remarque : *si vous ne fournissez pas de nom dans la méthode* `SharedObject.getLocal()`*, Flash Player nomme le fichier undefined.sol.*

Par défaut, Flash peut enregistrer des objets SharedObject persistants localement, d'une taille maximale de 100 Ko par domaine. Cette valeur est configurable par l'utilisateur. Lorsque l'application tente d'enregistrer un objet partagé d'une taille supérieure à 100 Ko, Flash Player affiche la boîte de dialogue Stockage local, qui permet à l'utilisateur d'autoriser ou de refuser une augmentation du volume de stockage local pour le domaine qui demande l'accès.

Spécification d'un chemin

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le paramètre facultatif *pathname* permet de spécifier l'emplacement du fichier SharedObject. Ce fichier doit être un sous-répertoire du répertoire SharedObject de ce domaine. Par exemple, si vous demandez une application sur l'hôte local et spécifiez ce qui suit :

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

Flash Player écrit le fichier SharedObject dans le répertoire /#localhost (ou /localhost si l'application est hors ligne). Cela s'avère utile si vous souhaitez plusieurs applications sur le client pour pouvoir accéder au même objet partagé. Dans ce cas, le client peut exécuter deux applications Flex, qui spécifient un chemin vers l'objet partagé qui est la racine du domaine ; le client peut ensuite accéder au même objet partagé à partir des deux applications. Pour partager des données entre plusieurs applications sans persistance, vous pouvez utiliser l'objet LocalConnection.

Si vous spécifiez un répertoire inexistant, Flash Player ne crée pas de fichier SharedObject.

Ajout de données à un objet partagé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous ajoutez des données à un fichier *.sol d'objet SharedObject à l'aide de sa propriété `data`. Pour ajouter de nouvelles données à l'objet partagé, utilisez la syntaxe suivante :

```
sharedObject_name.data.variable = value;
```

L'exemple suivant ajoute les propriétés `userName`, `itemNumbers` et `adminPrivileges` et leurs valeurs à un objet SharedObject :

```
public var currentUserName:String = "Reiner";
public var itemsArray:Array = new Array(101,346,483);
public var currentUserIsAdmin:Boolean = true;
mySO.data.userName = currentUserName;
mySO.data.itemNumbers = itemsArray;
mySO.data.adminPrivileges = currentUserIsAdmin;
```

Après avoir affecté des valeurs à la propriété `data`, vous devez demander à Flash Player les écrire dans le fichier SharedObject. Pour forcer Flash Player à écrire les valeurs dans le fichier SharedObject, utilisez la méthode `SharedObject.flush()`, comme suit :

```
mySO.flush();
```

Si vous n'appelez pas la méthode `SharedObject.flush()`, Flash Player écrit les valeurs dans le fichier à la fermeture de l'application. Cependant, cette méthode ne permet pas à l'utilisateur d'augmenter l'espace disponible nécessaire à Flash Player pour enregistrer les données si la taille de ces dernières est supérieure aux paramètres par défaut. La meilleure pratique consiste donc à appeler `SharedObject.flush()`.

Si vous utilisez la méthode `flush()` pour écrire des objets partagés sur le disque dur de l'ordinateur, vous devez vérifier avec soin si l'utilisateur a explicitement désactivé le stockage local à l'aide du Gestionnaire de paramètres de Flash Player (www.macromedia.com/support/documentation/fr/flashplayer/help/settings_manager07.html), comme illustré dans cet exemple :

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

Stockage d'objets dans les objets partagés

Vous pouvez stocker des objets simples comme Arrays ou Strings dans la propriété `data` de l'objet SharedObject.

L'exemple suivant est une class ActionScript qui définit des méthodes contrôlant l'interaction avec l'objet partagé. Ces méthodes permettent à l'utilisateur d'ajouter et de supprimer des objets de l'objet partagé. Cette classe stocke une collection ArrayCollection qui contient des objets simples.

```
package {
    import mx.collections.ArrayCollection;
    import flash.net.SharedObject;

    public class LSOHandler {

        private var mySO:SharedObject;
        private var ac:ArrayCollection;
        private var lsoType:String;

        // The parameter is "feeds" or "sites".
        public function LSOHandler(s:String) {
            init(s);
        }

        private function init(s:String):void {
            ac = new ArrayCollection();
            lsoType = s;
            mySO = SharedObject.getLocal(lsoType);
            if (getObjects()) {
```

```
                ac = getObjects();
            }
        }

        public function getObjects():ArrayCollection {
            return mySO.data[lsoType];
        }

        public function addObject(o:Object):void {
            ac.addItem(o);
            updateSharedObjects();
        }

        private function updateSharedObjects():void {
            mySO.data[lsoType] = ac;
            mySO.flush();
        }
    }

}
```

L'application Flex suivante crée une occurrence de la classe ActionScript pour chacun des types d'objets partagés dont elle a besoin. Elle appelle ensuite des méthodes sur cette classe lorsque l'utilisateur ajoute ou supprime des blogs ou des URL de site.

```
<?xml version="1.0"?>
<!-- lsos/BlogAggregator.mxml -->
<mx:Application
    xmlns:local="*"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    creationComplete="initApp()"
    backgroundColor="#ffffff"
>
    <mx:Script>
        <![CDATA[
        import mx.collections.ArrayCollection;
        import mx.utils.ObjectUtil;
        import flash.net.SharedObject;

        [Bindable]
        public var welcomeMessage:String;

        [Bindable]
        public var localFeeds:ArrayCollection = new ArrayCollection();

        [Bindable]
        public var localSites:ArrayCollection = new ArrayCollection();

        public var lsofeeds:LSOHandler;
        public var lsosites:LSOHandler;

        private function initApp():void {
            lsofeeds = new LSOHandler("feeds");
            lsosites = new LSOHandler("sites");

            if (lsofeeds.getObjects()) {
                localFeeds = lsofeeds.getObjects();
```

```
            }
            if (lsosites.getObjects()) {
                localSites = lsosites.getObjects();
            }
        }

        // Adds a new feed to the feeds DataGrid.
        private function addFeed():void {
            // Construct an object you want to store in the
            // LSO. This object can contain any number of fields.
            var o:Object = {name:ti1.text, url:ti2.text, date:new Date()};
            lsofeeds.addObject(o);

            // Because the DataGrid's dataProvider property is
            // bound to the ArrayCollection, Flex updates the
            // DataGrid when you call this method.
            localFeeds = lsofeeds.getObjects();

            // Clear the text fields.
            ti1.text = '';
            ti2.text = '';
        }

        // Removes feeds from the feeds DataGrid.
        private function removeFeed():void {
            // Use a method of ArrayCollection to remove a feed.
            // Because the DataGrid's dataProvider property is
            // bound to the ArrayCollection, Flex updates the
            // DataGrid when you call this method. You do not need
            // to update it manually.
            if (myFeedsGrid.selectedIndex > -1) {

localFeeds.removeItemAt(myFeedsGrid.selectedIndex);
             }
        }

        private function addSite():void {
            var o:Object = {name:ti3.text, date:new Date()};
            lsosites.addObject(o);
            localSites = lsosites.getObjects();
            ti3.text = '';
        }

        private function removeSite():void {
            if (mySitesGrid.selectedIndex > -1) {

localSites.removeItemAt(mySitesGrid.selectedIndex);
            }
        }

        ]]>
    </mx:Script>

    <mx:Label text="Blog aggregator" fontSize="28"/>

    <mx:Panel title="Blogs">
        <mx:Form id="blogForm">
```

```
            <mx:HBox>
                <mx:FormItem label="Name:">
                    <mx:TextInput id="ti1" width="100"/>
                </mx:FormItem>
                <mx:FormItem label="Location:">
                    <mx:TextInput id="ti2" width="300"/>
                </mx:FormItem>
                <mx:Button id="b1" label="Add Feed" click="addFeed()"/>
            </mx:HBox>

            <mx:FormItem label="Existing Feeds:">
                <mx:DataGrid
                    id="myFeedsGrid"
                    dataProvider="{localFeeds}"
                    width="400"
                />
            </mx:FormItem>
            <mx:Button id="b2" label="Remove Feed" click="removeFeed()"/>
        </mx:Form>
    </mx:Panel>

    <mx:Panel title="Sites">
        <mx:Form id="siteForm">
            <mx:HBox>
                <mx:FormItem label="Site:">
                    <mx:TextInput id="ti3" width="400"/>
                </mx:FormItem>
                <mx:Button id="b3" label="Add Site" click="addSite()"/>
            </mx:HBox>

            <mx:FormItem label="Existing Sites:">
                <mx:DataGrid
                    id="mySitesGrid"
                    dataProvider="{localSites}"
                    width="400"
                />
            </mx:FormItem>
            <mx:Button id="b4" label="Remove Site" click="removeSite()"/>
        </mx:Form>
    </mx:Panel>

</mx:Application>
```

Stockage des objets typés dans les objets partagés

Vous pouvez stocker des occurrences ActionScript typées dans des objets partagés. Pour ce faire, il vous suffit d'appeler la méthode `flash.net.registerClassAlias()` pour enregistrer la classe. Si vous créez une occurrence de votre classe et que vous la stockez dans les données membres de votre objet partagé, puis que vous lisez ce dernier, vous obtenez une occurrence typée. Par défaut, la propriété `objectEncoding` de l'objet SharedObject prend en charge le codage AMF3 et décompresse votre occurrence stockée à partir de l'objet SharedObject ; l'occurrence stockée conserve le même type spécifié lorsque vous avez appelé la méthode `registerClassAlias()`.

(iOS uniquement) Empêcher la sauvegarde sur le cloud des objets partagés locaux

Adobe AIR 3.7 et ultérieur, iOS uniquement

Vous pouvez définir la propriété `SharedObject.preventBackup` de manière à contrôler si les objets locaux partagés sont sauvegardés sur le service iOS de sauvegarde sur le cloud. Apple demande de procéder de cette manière pour tout contenu qu'il est possible de générer ou télécharger à nouveau, mais qui est nécessaire au bon fonctionnement de votre application lors d'une utilisation hors ligne.

Création de plusieurs objets partagés

Vous pouvez créer plusieurs objets partagés pour la même application Flex. Pour ce faire, affectez à chacun d'entre eux un nom d'occurrence différent, comme indiqué dans l'exemple suivant :

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

Cette action entraîne la création des fichiers preferences.sol et history.sol file dans le répertoire local de l'application Flex.

Création d'un SharedObject sécurisé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez un SharedObject local ou distant à l'aide de `getLocal()` ou `getRemote()`, un paramètre facultatif nommé `secure` détermine si l'accès à l'objet partagé se limite aux fichiers SWF diffusés via une connexion HTTPS. Si ce paramètre a la valeur `true` et que votre fichier SWF est diffusé sur HTTPS, Flash Player crée un nouvel objet sécurisé ou obtient une référence à un objet partagé sécurisé existant. Cet objet partagé sécurisé peut uniquement être lu ou écrit par des fichiers SWF reçus via des connexions HTTPS appelant `SharedObject.getLocal()` avec le paramètre secure réglé sur `true`. Si ce paramètre a la valeur `false` et que votre fichier SWF est diffusé sur HTTPS, Flash Player crée un nouvel objet partagé ou obtient une référence à un objet partagé existant.

Cet objet partagé peut être lu ou écrit par des fichiers SWF reçus via des connexions autres que HTTPS. Si votre fichier SWF est diffusé via une connexion autre que HTTPS et que vous essayez de régler ce paramètre sur `true`, la création d'un objet partagé (ou l'accès à un objet partagé sécurisé précédemment créé) échoue, une erreur est renvoyée et l'objet partagé devient `null`. Si vous tentez d'exécuter l'extrait de code suivant à partir d'un connexion non HTTPS, la méthode `SharedObject.getLocal()` renvoie une erreur :

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

Quelle que soit la valeur de ce paramètre, les objets partagés créés sont comptabilisés dans la quantité d'espace disque total autorisée pour un domaine.

Affichage du contenu d'un objet partagé

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Des valeurs sont stockées dans un objet partagé, au sein de la propriété `data`. Vous pouvez passer en boucle chaque valeur d'une occurrence d'objet partagé à l'aide de la boucle `for..in`, comme le montre l'exemple suivant :

```
var so:SharedObject = SharedObject.getLocal("test");
so.data.hello = "world";
so.data.foo = "bar";
so.data.timezone = new Date().timezoneOffset;
for (var i:String in so.data)
{
    trace(i + ":\t" + so.data[i]);
}
```

Destruction d'objets partagés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour détruire un objet SharedObject sur le client, faites appel à la méthode `SharedObject.clear()`. Cette action ne détruit pas les répertoires du chemin par défaut pour les objets partagés de l'application.

L'exemple suivant supprime le fichier SharedObject du client :

```
public function destroySharedObject():void {
    mySO.clear();
}
```

Exemple SharedObject

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant illustre le stockage d'objets simples, tel un objet Date, dans un objet SharedObject sans avoir à sérialiser et désérialiser manuellement ces objets.

L'exemple suivant commence par un souhait de bienvenue en tant que nouveau visiteur. Lorsque vous cliquez sur Déconnexion, l'application stocke la date actuelle dans un objet partagé. La prochaine fois que vous lancez cette application ou que vous actualisez la page, l'application vous souhaite la bienvenue avec un rappel de l'heure à laquelle vous vous êtes déconnecté.

Pour voir l'application en action, lancez-la, cliquez sur Déconnexion, puis réactualisez la page. L'application affiche la date et l'heure auxquelles vous avez cliqué sur le bouton Déconnexion lors de votre précédente visite. Vous pouvez supprimer à tout moment les informations stockées en cliquant sur le bouton Supprimer le LSO.

```
<?xml version="1.0"?>
<!-- lsos/WelcomeMessage.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" initialize="initApp()">
  <mx:Script><![CDATA[
  public var mySO:SharedObject;
  [Bindable]
  public var welcomeMessage:String;

  public function initApp():void {
     mySO = SharedObject.getLocal("mydata");
     if (mySO.data.visitDate==null) {
        welcomeMessage = "Hello first-timer!"
     } else {
        welcomeMessage = "Welcome back. You last visited on " +
           getVisitDate();
     }
  }

  private function getVisitDate():Date {
     return mySO.data.visitDate;
  }

  private function storeDate():void {
     mySO.data.visitDate = new Date();
     mySO.flush();
  }

  private function deleteLSO():void {
     // Deletes the SharedObject from the client machine.
     // Next time they log in, they will be a 'first-timer'.
     mySO.clear();
  }

  ]]></mx:Script>
  <mx:Label id="label1" text="{welcomeMessage}"/>
  <mx:Button label="Log Out" click="storeDate()"/>
  <mx:Button label="Delete LSO" click="deleteLSO()"/>
</mx:Application>
```

Stockage local chiffré

La classe EncryptedLocalStore (ELS) fournit un mécanisme de stockage local chiffré que vous pouvez utiliser comme mémoire cache pour stocker les données privées d'une application. Les données du magasin local chiffré ne peuvent pas être partagées entre les applications. L'objectif du magasin local chiffré est de permettre à une application de stocker les éléments facilement recréés tels que les informations d'identification et autres informations privées. Les données du magasin local chiffré ne doivent pas être considérées comme étant permanentes, comme nous l'indiquons dans les rubriques « Restrictions du magasin local chiffré » et « Recommandations d'utilisation » ci-dessous.

Remarque : *outre le magasin local chiffré, AIR assure également le chiffrement du contenu stocké dans les bases de données SQL. Pour plus d'informations, voir la section «* *Utilisation du chiffrement avec les bases de données SQL* » *à la page 787*.

Vous pouvez utiliser le magasin local chiffré pour placer en mémoire cache les informations à sécuriser, telles que les informations d'identification de connexion aux services Web. Le magasin local chiffré est adapté au stockage d'informations confidentielles. Il ne protège toutefois pas les données des autres processus exécutés à l'aide du même compte utilisateur. Il n'est par conséquent pas adapté à la protection des données d'applications confidentielles telles que DRM ou les clés de chiffrement.

Sur les plates-formes de bureau, AIR utilise DPAPI sous Windows, KeyChain sous Mac OS et iOS, et KeyRing ou KWallet sous Linux pour associer le magasin local chiffré à chaque application et chaque utilisateur. Le magasin local chiffré utilise le chiffrement AES-CBC 128 bits.

Sur Android, les données enregistrées par la classe EncryptedLocalStorage ne sont pas chiffrées. Elles sont protégées par la sécurité de niveau utilisateur fournie par le système d'exploitation. Le système d'exploitation Android affecte à chaque application un ID utilisateur distinct. Les applications peuvent uniquement accéder à leurs propres fichiers et aux fichiers créés dans des emplacements publics (tels que la carte de stockage amovible). Notez que sur les périphériques Android associés à une racine, les applications s'exécutant avec des privilèges racines PEUVENT accéder aux fichiers d'autres applications. Sur ce type de périphérique, le magasin de stockage local ne fournit pas un niveau élevé de protection de données.

Les informations stockées dans le magasin local chiffré sont réservées au contenu d'application AIR dans le sandbox de sécurité de l'application.

Une version mise à jour d'une application AIR conserve un accès à toute donnée existante stockée dans le magasin local chiffré, sauf si :

- Lors de l'ajout des éléments, le paramètre `stronglyBound` était défini sur `true`.
- La version existante et la version mise à jour sont toutes deux publiées avant AIR 1.5.3 et la mise à jour est dotée d'une signature de migration.

Restrictions du magasin local chiffré

Les données stockées dans le magasin local chiffré sont protégées par les informations d'identification du compte sur le système d'exploitation de l'utilisateur. Sauf s'il se connecte sous le nom de cet utilisateur, aucun autre utilisateur ne peut accéder aux données du magasin. Les données ne sont toutefois pas protégées contre les accès par d'autres applications exécutées par un utilisateur authentifié.

Puisque l'utilisateur doit être authentifié pour que ces attaques aboutissent, les données privées de ce dernier demeurent protégées, à moins que le compte utilisateur en tant que tel ne soit plus sécurisé. Toutefois, les données que votre application ne souhaite pas partager avec d'autres utilisateurs, telles que les clés d'obtention de licence ou de gestion des droits numériques, ne sont pas sécurisées. L'ESL n'est donc pas adapté au stockage d'informations de ce type. Il convient uniquement au stockage des données privées d'un utilisateur, telles que les mots de passe.

Les données stockées dans l'ELS risquent d'être perdues pour diverses raisons. L'utilisateur pourrait par exemple désinstaller l'application et supprimer le fichier chiffré. L'identifiant d'éditeur pourrait également être modifié suite à une mise à jour. L'ELS doit de ce fait être traité comme une mémoire cache privée et non un emplacement de stockage de données permanent.

L'utilisation du paramètre `stronglyBound` étant déconseillée, ne le définissez pas sur `true`. Définir ce paramètre sur `true` ne constitue pas une mesure de protection complémentaire des données. L'accès aux données est par ailleurs perdu lorsque l'application est mise à jour, même si l'identifiant d'éditeur ne change pas.

Les performances du magasin local chiffré risquent d'être plus lentes si les données stockées dépassent 10 Mo.

Lorsque vous désinstallez une application AIR, le programme de désinstallation ne supprime pas les données stockées dans le magasin local chiffré.

Recommandations d'utilisation

Les recommandations d'utilisation de l'ELS sont les suivantes :

- Stockez dans l'ELS les données utilisateur confidentielles telles que les mots de passe (définissez le paramètre stronglyBound sur false).
- Ne stockez pas dans l'ELS des données secrètes d'applications telles que les clés DRM ou les jetons de licence..
- Intégrez dans l'application une technique de recréation des données stockées dans l'ELS en cas de perte des données stockées dans ce dernier. Vous pouvez, par exemple, inviter l'utilisateur à saisir à nouveau les informations d'identification du compte si besoin est.
- N'utilisez pas le paramètre `stronglyBound`.
- Si vous définissez `stronglyBound` sur `true`, n'effectuez pas la migration des éléments stockés lors d'une mise à jour. Créez à nouveau les données une fois la mise à jour terminée.
- Ne stockez que des volumes relativement faibles de données. Stockez les volumes plus importants de données dans une base de données SQL AIR avec chiffrement.

Voir aussi

flash.data.EncryptedLocalStore

Ajout de données au magasin local chiffré

La méthode statique `setItem()` de la classe EncryptedLocalStore permet de stocker des données dans le magasin local. Les données sont stockées dans une table de hachage, dans laquelle les chaînes font office de clés et les données stockées de tableaux d'octets.

Par exemple, le code suivant stocke une chaîne dans le magasin local chiffré :

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

Le troisième paramètre de la méthode `setItem()`, `stronglyBound`, est facultatif. Si ce paramètre est défini sur `true`, le magasin local chiffré lie l'élément stocké aux bits et à la signature numérique de l'application AIR qui effectue le stockage :

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

Si un élément est stocké alors que `stronglyBound` est défini sur `true`, les appels suivants de `getItem()` n'aboutissent que si l'application AIR à l'origine de l'appel est identique à l'application qui effectue le stockage (en d'autres termes, si aucune donnée des fichiers résidant dans le répertoire de l'application n'a été modifiée). Si l'application AIR à l'origine de l'appel n'est pas identique à l'application de stockage, une exception Error est renvoyée lorsque vous appelez `getItem()` pour un élément à liaison forte. Si vous mettez à jour votre application, elle n'est plus en mesure de lire les données à liaison forte précédemment écrites dans le magasin local chiffré. La définition de `stronglyBound` sur `true` sur les périphériques mobiles est ignorée ; ce paramètre est toujours considéré comme `false`.

Si le paramètre `stronglyBound` est défini sur `false` (valeur par défaut), il suffit que l'identifiant d'éditeur reste identique pour que l'application puisse lire les données. Les bits de l'application peuvent changer (sous réserve d'être signés par le même éditeur), mais ne doivent pas nécessairement correspondre exactement aux bits de l'application qui a stocké les données. Les applications mises à jour dotées du même identifiant d'éditeur que l'application originale continuent à accéder aux données.

Remarque : *en pratique, définir `stronglyBound` sur `true` n'assure aucune protection complémentaire des données. Un utilisateur « malveillant » pourrait en effet modifier une application en vue d'accéder aux éléments stockés dans l'ELS. Que `stronglyBound` soit défini sur `true` ou `false`, les données sont par ailleurs aussi bien protégées contre les menaces externes émanant de personnes autres que les utilisateurs. De ce fait, il est déconseillé de définir `stronglyBound` sur `true`.*

Accès aux données stockées dans le magasin local chiffré

Adobe AIR 1.0 et les versions ultérieures

Pour extraire une valeur du magasin local chiffré, utilisez la méthode `EncryptedLocalStore.getItem()`, comme dans l'exemple suivant :

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

Suppression de données du magasin local chiffré

Adobe AIR 1.0 et les versions ultérieures

Pour supprimer une valeur du magasin local chiffré, utilisez la méthode `EncryptedLocalStore.removeItem()`, comme dans l'exemple suivant :

```
EncryptedLocalStore.removeItem("firstName");
```

Pour supprimer toutes les valeurs du magasin local chiffré, appelez la méthode `EncryptedLocalStore.reset()`, comme dans l'exemple suivant :

```
EncryptedLocalStore.reset();
```

Chapitre 40 : Utilisation des bases de données SQL locales dans AIR

Adobe AIR 1.0 et les versions ultérieures

Adobe® AIR® permet de créer et d'utiliser des bases de données SQL locales. Le moteur d'exécution inclut un moteur de base de données SQL avec prise en charge de nombreuses fonctionnalités SQL standard, à l'aide du système de base de données SQLite open source. Une base de données SQL locale peut être utilisée pour le stockage des données persistantes locales. Par exemple, elle peut servir pour les données d'application, les paramètres utilisateur d'application, des documents ou tout autre type de données que votre application doit enregistrer localement.

A propos des bases de données SQL locales

Adobe AIR 1.0 et les versions ultérieures

Pour obtenir une explication rapide de l'utilisation de bases de données SQL et des exemples de code correspondants, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Utilisation asynchrone d'une base de données SQL locale (Flex)
- Utilisation asynchrone d'une base de données SQL locale (Flex)
- Utilisation d'une base de données chiffrée (Flex)
- Utilisation asynchrone d'une base de données SQL locale (Flash)
- Utilisation synchrone d'une base de données SQL locale (Flash)
- Utilisation d'une base de données chiffrée (Flash)

Adobe AIR comprend un moteur de base de données relationnelle de type SQL qui fonctionne au sein du moteur d'exécution. Les données sont stockées localement dans des fichiers de bases de données dans l'ordinateur sur lequel l'application AIR s'exécute (par exemple, dans le disque dur de l'ordinateur). L'exécution de la base de données et le stockage des fichiers de données s'effectuant localement, une application AIR peut utiliser une base de données qu'une connexion réseau soit disponible ou non. Le moteur de base de données SQL locale du moteur d'exécution constitue ainsi un mécanisme pratique pour le stockage des données d'application locales persistantes, en particulier si vous maîtrisez les bases de données relationnelles et SQL.

Cas d'utilisation des bases de données SQL locales

Adobe AIR 1.0 et les versions ultérieures

La fonctionnalité de base de données SQL locale de l'application AIR peut répondre à tout objectif de stockage des données d'application sur l'ordinateur local d'un utilisateur. Adobe AIR inclut plusieurs mécanismes de stockage local des données, chacun présentant des avantages distincts. Voici quelques utilisations possibles d'une base de données SQL locale dans votre application AIR :

- Avec une application orientée données (par exemple, un carnet d'adresses), une base de données peut être utilisée pour stocker les données de l'application principale.

- Avec une application orientée documents, dans laquelle les utilisateurs créent des documents à enregistrer et éventuellement à partager, chaque document peut être enregistré sous forme de fichier de base de données, à l'emplacement désigné par l'utilisateur. (Notez, toutefois, que si la base de données n'est pas chiffrée, toute application AIR peut ouvrir le fichier de bases de données. Le chiffrement est donc recommandé pour tous les documents potentiellement sensibles.)
- Avec une application réseau, il est possible d'utiliser une base de données pour stocker un cache local des données d'application ou pour stocker temporairement des données lorsque la connexion réseau n'est pas disponible. Un mécanisme de synchronisation peut dans ce cas être créé pour synchroniser la base de données locale avec le magasin de données en réseau.
- Quelle que soit l'application, une base de données peut être utilisée pour stocker les paramètres d'application des utilisateurs individuels, tels que les informations relatives aux applications ou aux options des utilisateurs, comme la taille et la position de la fenêtre.

Voir aussi

Christophe Coenraets : Employee Directory on AIR for Android (disponible en anglais uniquement)

Raymond Camden : jQuery and AIR - Moving from web page to application (disponible en anglais uniquement)

A propos des fichiers de bases de données et des bases de données AIR

Adobe AIR 1.0 et les versions ultérieures

Une base de données SQL locale Adobe AIR individuelle est stockée sous la forme d'un seul fichier dans le système de fichiers de l'ordinateur. Le moteur de base de données SQL du moteur d'exécution gère la création et le formatage des fichiers de base de données, de même que la manipulation et la récupération des données dans un fichier de base de données. Le moteur d'exécution ne spécifie pas comment ni où les données de la base de données sont stockées dans le système de fichiers, mais chaque base de données est stockée dans son intégralité dans un seul fichier. Vous spécifiez l'emplacement de stockage du fichier de base de données dans le système de fichiers. Une même application AIR peut accéder à une ou plusieurs bases de données distinctes (c'est-à-dire à des fichiers de base de données distincts). Le moteur d'exécution stockant chaque base de données sous forme d'un seul fichier dans le système de fichiers, vous pouvez localiser votre base de données selon vos besoins, en fonction de la conception de votre application et des contraintes d'accès aux fichiers du système d'exploitation. Chaque utilisateur peut disposer d'un fichier de base de données distinct pour ses données spécifiques, ou tous les utilisateurs de l'application peuvent accéder à un fichier de bases de données sur un ordinateur dans le cas de données partagées. Les données étant stockées localement sur un seul ordinateur, elles ne sont pas automatiquement partagées avec les utilisateurs d'autres ordinateurs. Le moteur de la base de données SQL locale ne permet pas d'exécuter des instructions SQL sur une base de données distante ou basée sur un serveur.

A propos des bases de données relationnelles

Adobe AIR 1.0 et les versions ultérieures

Une base de données relationnelle est un mécanisme de stockage (et de récupération) des données sur un ordinateur. Les données sont organisées en tables : les lignes représentent les enregistrements ou les éléments, et les colonnes (parfois appelées « champs ») divisent chaque enregistrement en valeurs individuelles. Par exemple, une application de carnet d'adresses peut contenir une table nommée « amis ». Chaque ligue de la table représente alors un ami stocké dans la base de données. Les colonnes de cette table représentent des données telles que le prénom, le nom, la date de naissance, etc. Pour chaque ligne de la table, la base de données stocke une valeur distincte dans chaque colonne.

Les bases de données relationnelles sont conçues pour stocker des données complexes, chaque élément étant associé ou relié à des éléments d'un autre type. Dans une base de données relationnelle, toute donnée présentant une relation « un à plusieurs » (où un seul enregistrement peut être relié à plusieurs enregistrements de types différents) doit être répartie parmi les différentes tables. Par exemple, si vous souhaitez stocker plusieurs numéros de téléphone pour chaque ami dans votre application de carnet d'adresses, il s'agit d'une relation « un à plusieurs ». La table « amis » contiendrait alors toutes les informations personnelles de chaque ami. Une table distincte « numéros de téléphone » contiendrait tous les numéros de téléphone de tous les amis.

Outre le stockage des données de vos amis et de leurs numéros de téléphone, chaque table a besoin d'un élément de données qui lui permette de relier les deux tables (pour mettre en correspondance les enregistrements individuels des amis et leurs numéros de téléphone). Ces données sont appelées clé primaire (identifiant unique qui différencie chaque ligne de la table des autres lignes de cette même table). La clé primaire peut être une « clé naturelle », c'est-à-dire un élément de données qui différencie naturellement chaque enregistrement d'une table. Dans le cas de la table « amis », si vous savez qu'aucun de vos amis n'a la même date de naissance, vous pouvez utiliser la colonne date de naissance comme clé primaire (clé naturelle) de cette table. S'il n'existe pas de clé naturelle, vous pouvez créer une colonne de clé primaire distincte telle que « ID ami » (valeur artificielle utilisée par l'application pour différencier les lignes).

L'utilisation d'une clé primaire permet de définir les relations existant entre plusieurs tables. Supposons par exemple que la colonne « ID ami » de la table « amis » contienne un numéro unique pour chaque ligne (chaque ami). La table reliée « numéros de téléphone » peut être structurée sur deux colonnes : l'une contenant l'identifiant de l'ami à qui le numéro de téléphone appartient, l'autre contenant le numéro de téléphone lui-même. Ainsi, quel que soit le nombre de numéros de téléphone d'un ami, tous peuvent être stockés dans la table « numéros de téléphone » et reliés à l'ami associé via la clé primaire « ID ami ». Lorsque la clé primaire d'une table est utilisée dans une table reliée pour spécifier la connexion entre les enregistrements, la valeur de la table reliée est appelée clé étrangère. Contrairement à de nombreuses bases de données, le moteur de base de données locale de l'application AIR ne vous permet pas de créer des contraintes de clé étrangère. Ces contraintes vérifient automatiquement qu'une valeur de clé étrangère mise à jour ou insérée correspond à une ligne de la table de clés primaires. Les relations entre les clés étrangères sont toutefois un élément important de la structure d'une base de données relationnelle, et les clés étrangères doivent être utilisées lors de la création de relations entre les tables de votre base de données.

A propos de SQL

Adobe AIR 1.0 et les versions ultérieures

Le langage SQL (Structured Query Language) est utilisé avec les bases de données relationnelles pour manipuler et récupérer les données. SQL est davantage un langage descriptif qu'un langage procédural. Au lieu de donner à l'ordinateur des instructions sur la façon dont les données doivent être récupérées, une instruction SQL décrit le jeu de données désiré. Le moteur de base de données détermine lui comment récupérer ces données.

Le langage SQL a été normalisé par l'institut ANSI (American National Standards Institute). La base de données SQL locale d'Adobe AIR prend en charge la plupart des standards SQL-92.

Pour consulter des descriptions spécifiques du langage SQL pris en charge dans Adobe AIR, voir « Prise en charge de SQL dans les bases de données locales » à la page 1154.

A propos des classes de base de données SQL

Adobe AIR 1.0 et les versions ultérieures

Pour travailler avec des bases de données SQL locales dans ActionScript 3.0, vous utilisez les occurrences des classes suivantes du package flash.data :

Classe	Description
flash.data.SQLConnection	Permet de créer et d'ouvrir des bases de données (fichiers de base de données), de même que des méthodes pour effectuer des opérations au niveau des bases de données et pour contrôler leurs transactions.
flash.data.SQLStatement	Représente une seule instruction SQL (une unique requête ou commande) exécutée sur une base de données, avec la définition du texte de l'instruction et les valeurs des paramètres.
flash.data.SQLResult	Permet de récupérer des informations ou les résultats provenant de l'exécution d'une instruction, par exemple le résultat des lignes provenant d'une instruction `SELECT`, le nombre de lignes affectées par une instruction `UPDATE` ou `DELETE`, etc.

Pour obtenir les informations du schéma décrivant la structure d'une base de données, utilisez les classes suivantes du package flash.data :

Classe	Description
flash.data.SQLSchemaResult	Sert de conteneur aux résultats du schéma de la base de données générés par un appel à la méthode `SQLConnection.loadSchema()`.
flash.data.SQLTableSchema	Fournit des informations décrivant une table spécifique dans une base de données.
flash.data.SQLViewSchema	Fournit des informations décrivant une vue spécifique dans une base de données.
flash.data.SQLIndexSchema	Fournit des informations décrivant une colonne spécifique dans une table ou une vue d'une base de données.
flash.data.SQLTriggerSchema	Fournit des informations décrivant un déclencheur spécifique dans une base de données.

Les autres classes du package flash.data fournissent des constantes utilisées avec les classes SQLConnection et SQLColumnSchema :

Classe	Description
flash.data.SQLMode	Définit un ensemble de constantes représentant les valeurs possibles du paramètre `openMode` des méthodes `SQLConnection.open()` et `SQLConnection.openAsync()`.
flash.data.SQLColumnNameStyle	Définit un ensemble de constantes représentant les valeurs possibles de la propriété `SQLConnection.columnNameStyle`.
flash.data.SQLTransactionLockType	Définit un ensemble de constantes représentant les valeurs possibles du paramètre option de la méthode `SQLConnection.begin()`.
flash.data.SQLCollationType	Défini un ensemble de constantes représentants les valeurs possibles de la propriété `SQLColumnSchema.defaultCollationType` et du paramètre `defaultCollationType` du constructeur `SQLColumnSchema()`.

De plus, les classes suivantes du package flash.events représentent les événements (et les constantes prises en charge) que vous utilisez :

Classe	Description
flash.events.SQLEvent	Définit les événements distribués par une occurrence de SQLConnection ou SQLStatement lorsque l'une de ses opérations s'exécute avec succès. Chaque opération est associée à une constante de type d'événement définie dans la classe SQLEvent.
flash.events.SQLErrorEvent	Définit l'événement distribué par une occurrence de SQLConnection ou SQLStatement lorsque l'une de ses opérations provoque une erreur.
flash.events.SQLUpdateEvent	Définit l'événement distribué par une occurrence de SQLConnection lorsque les données d'une table de l'une de ses bases de données connectées changent du fait de l'exécution d'une instruction SQL `INSERT`, `UPDATE` ou `DELETE`.

Enfin, les classes suivantes du package flash.errors fournissent des informations sur les erreurs des opérations de base de données :

Classe	Description
flash.errors.SQLError	Fournit des informations sur une erreur d'opération de base de données, y compris l'opération tentée et la cause de son échec.
flash.errors.SQLErrorOperation	Définit un ensemble de constantes représentant les valeurs possibles de la propriété `operation` de la classe SQLError, indiquant l'opération de base de données ayant provoqué une erreur.

A propos des modes d'exécution synchrone et asynchrone

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous écrivez du code pour travailler avec une base de données SQL locale, vous spécifiez que les opérations de cette base de données doivent s'exécuter dans l'un des deux modes d'exécution suivants : asynchrone ou synchrone. En général, les exemples de code montrent comment effectuer chaque opération des deux manières. Vous pouvez donc utiliser l'exemple répondant le mieux à vos besoins.

En mode d'exécution asynchrone, vous envoyez une instruction au moteur d'exécution et ce dernier distribue un événement lorsque l'opération demandée se termine ou échoue. Vous commencez par indiquer au moteur de base de données d'effectuer une opération. Celui-ci travaille en arrière-plan pendant que l'application poursuit son exécution. Enfin, lorsque l'opération est terminée (ou lorsqu'elle échoue), le moteur de base de données distribue un événement. Votre code, déclenché par l'événement, effectue les opérations consécutives. Cette approche présente un avantage important : le moteur d'exécution effectue les opérations de base de données en arrière-plan pendant que le code de l'application principale poursuit son exécution. Si l'opération de base de données prend un certain temps, l'exécution de l'application n'est pas interrompue. Plus important encore, l'utilisateur peut continuer à interagir avec elle sans que l'écran ne se fige. Toutefois, la rédaction du code d'opérations asynchrones peut se révéler plus complexe que la rédaction d'autres codes. Cette complexité survient généralement lorsque plusieurs opérations dépendantes doivent être réparties entre diverses méthodes d'écouteur d'événement.

De façon conceptuelle, il est plus simple de coder les opérations sous forme d'une seule séquence d'étapes (ensemble d'opérations synchrones) plutôt que sous forme d'un ensemble d'opérations divisées en plusieurs méthodes d'écouteur d'événement. Outre les opérations de base de données asynchrones, Adobe AIR vous permet également d'exécuter des opérations de base de données de façon synchrone. Dans ce mode, les opérations ne s'exécutent pas en arrière-plan mais dans la même séquence d'exécution que le reste du code de l'application. Vous indiquez au moteur de base de données d'effectuer une opération. Le code s'interrompt alors à ce stade pendant que le moteur de base de données effectue son travail. Lorsque l'opération est terminée, l'exécution se poursuit avec la ligne suivante de votre code.

L'exécution asynchrone ou synchrone des opérations est définie au niveau de l'occurrence de SQLConnection. L'utilisation d'une seule connexion de base de données ne permet pas d'exécuter certaines opérations ou instructions de façon synchrone et d'autres de façon asynchrone. Pour indiquer si une occurrence de SQLConnection fonctionne en mode d'exécution synchrone ou asynchrone, appelez une méthode SQLConnection pour ouvrir la base de données. Si vous appelez `SQLConnection.open()`, la connexion opère en mode d'exécution synchrone, et si vous appelez `SQLConnection.openAsync()`, le mode d'exécution asynchrone est utilisé. Lorsqu'une occurrence de SQLConnection est connectée à une base de données à l'aide de `open()` ou `openAsync()`, elle est figée en mode d'exécution synchrone ou asynchrone, sauf si vous fermez et rouvrez la connexion à la base de données.

Chaque mode d'exécution a ses propres avantages. Bien que similaires dans la plupart de leurs aspects, chaque mode présente des différences que vous devez connaître pour les exploiter correctement. Pour plus d'informations et pour obtenir des suggestions quant au choix de chaque mode, voir la section « Utilisation des opérations de base de données synchrones et asynchrones » à la page 782.

Création et modification d'une base de données

Adobe AIR 1.0 et les versions ultérieures

Pour que votre application puisse ajouter ou récupérer des données, elle doit pouvoir accéder à une base de données avec des tables définies. Cette section décrit la création d'une base de données et de la structure de ses données. Bien que moins fréquemment utilisées que l'insertion et la récupération de données, ces tâches sont nécessaires pour la plupart des applications.

Voir aussi

Mind the Flex : Updating an existing AIR database (disponible en anglais uniquement)

Création d'une base de données

Adobe AIR 1.0 et les versions ultérieures

Pour créer un fichier de base de données, vous commencez par créer une occurrence de SQLConnection. Vous appelez sa méthode `open()` pour l'ouvrir en mode d'exécution synchrone, ou sa méthode `openAsync()` pour l'ouvrir en mode d'exécution asynchrone. Les méthodes `open()` et `openAsync()` sont utilisées pour ouvrir une connexion à une base de données. Si vous transmettez une occurrence de File faisant référence à un emplacement de fichier non existant pour le paramètre `reference` (premier paramètre), la méthode `open()` ou `openAsync()` crée un fichier de base de données à cet emplacement et ouvre une connexion à la nouvelle base de données.

Quelle que soit la méthode appelée pour créer une base de données, `open()` ou `openAsync()`, le nom du fichier de la base de données peut être tout nom de fichier valide, avec n'importe quelle extension. Si vous appelez la méthode `open()` ou `openAsync()` en définissant `null` pour le paramètre `reference`, une nouvelle base de données en mémoire est créée et non un fichier de base de données sur disque.

Le code suivant montre le processus de création d'un fichier de base de données (nouvelle base de données) en mode d'exécution asynchrone. Dans ce cas, le fichier de base de données est enregistré dans le « Pointage vers le répertoire de stockage d'une application » à la page 697, sous le nom « DBSample.db » :

```
import flash.data.SQLConnection;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

conn.openAsync(dbFile);

function openHandler(event:SQLEvent):void
{
    trace("the database was created successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                conn.openAsync(dbFile);
            }

            private function openHandler(event:SQLEvent):void
            {
                trace("the database was created successfully");
            }

            private function errorHandler(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Remarque : *bien que la classe File vous permette de pointer vers un chemin de fichier natif déterminé, les applications risquent alors de ne pas fonctionner sur toutes les plates-formes. Par exemple, le chemin C:\Documents and Settings\joe\test.db ne fonctionne que sous Windows. C'est pourquoi il est recommandé d'utiliser les propriétés statiques de la classe File, telles que* `File.applicationStorageDirectory`*, ainsi que la méthode* `resolvePath()` *(comme illustré par l'exemple précédent). Pour plus d'informations, voir «* *Chemin des objets File* *» à la page 693.*

Pour exécuter des opérations en mode synchrone, lorsque vous ouvrez une connexion à la base de données avec l'occurrence de SQLConnection, appelez la méthode `open()`. L'exemple suivant montre comment créer et ouvrir une occurrence de SQLConnection qui exécute ses opérations de façon synchrone :

```
import flash.data.SQLConnection;
import flash.errors.SQLError;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

try
{
    conn.open(dbFile);
    trace("the database was created successfully");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.errors.SQLError;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                try
                {
                    conn.open(dbFile);
                    trace("the database was created successfully");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Création de tables de base de données

Adobe AIR 1.0 et les versions ultérieures

La création d'une table dans une base de données implique l'exécution d'une instruction SQL sur cette base de données, selon la procédure utilisée pour exécuter une instruction SQL telle que `SELECT`, `INSERT`, etc. Pour créer une table, vous utilisez une instruction `CREATE TABLE`, qui inclut les définitions des colonnes et les contraintes de la nouvelle table. Pour plus d'informations sur l'exécution d'instructions SQL, voir la section « Utilisation des instructions SQL » à la page 757.

L'exemple suivant montre la création d'une table nommée « employees » dans un fichier de base de données existant, en mode d'exécution asynchrone. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à une base de données.

```
import flash.data.SQLConnection;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

var createStmt:SQLStatement = new SQLStatement();
createStmt.sqlConnection = conn;

var sql:String =
    "CREATE TABLE IF NOT EXISTS employees (" +
    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "   firstName TEXT, " +
    "   lastName TEXT, " +
    "   salary NUMERIC CHECK (salary > 0)" +
    ")";
createStmt.text = sql;

createStmt.addEventListener(SQLEvent.RESULT, createResult);
createStmt.addEventListener(SQLErrorEvent.ERROR, createError);

createStmt.execute();

function createResult(event:SQLEvent):void
{
    trace("Table created");
}

function createError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                var createStmt:SQLStatement = new SQLStatement();
                createStmt.sqlConnection = conn;

                var sql:String =
                    "CREATE TABLE IF NOT EXISTS employees (" +
                    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "   firstName TEXT, " +
                    "   lastName TEXT, " +
                    "   salary NUMERIC CHECK (salary > 0)" +
                    ")";
                createStmt.text = sql;

                createStmt.addEventListener(SQLEvent.RESULT, createResult);
                createStmt.addEventListener(SQLErrorEvent.ERROR, createError);

                createStmt.execute();
            }

            private function createResult(event:SQLEvent):void
            {
                trace("Table created");
            }

            private function createError(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

L'exemple suivant illustre la création d'une table nommée « employees » dans un fichier de base de données existant, en mode d'exécution synchrone. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à une base de données.

```
import flash.data.SQLConnection;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

var createStmt:SQLStatement = new SQLStatement();
createStmt.sqlConnection = conn;

var sql:String =
    "CREATE TABLE IF NOT EXISTS employees (" +
    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
    "   firstName TEXT, " +
    "   lastName TEXT, " +
    "   salary NUMERIC CHECK (salary > 0)" +
    ")";
createStmt.text = sql;

try
{
    createStmt.execute();
    trace("Table created");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                var createStmt:SQLStatement = new SQLStatement();
                createStmt.sqlConnection = conn;

                var sql:String =
                    "CREATE TABLE IF NOT EXISTS employees (" +
                    "   empId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "   firstName TEXT, " +
                    "   lastName TEXT, " +
                    "   salary NUMERIC CHECK (salary > 0)" +
                    ")";
                createStmt.text = sql;

                try
                {
                    createStmt.execute();
                    trace("Table created");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Manipulation des données de bases de données SQL

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous utilisez des bases de données SQL locales, vous effectuez certaines tâches courantes. Ces tâches incluent la connexion à une base de données et l'ajout et la récupération de données dans ses tables. Vous devez également être conscient(e) de différents problèmes lorsque vous effectuez ces tâches, par exemple l'utilisation des types de données et la gestion des erreurs.

Notez également que plusieurs tâches de base de données sont effectuées moins fréquemment mais doivent souvent l'être avant de vous attaquer à ces tâches plus courantes. Par exemple, pour pouvoir vous connecter à une base de données et récupérer des données dans l'une de ses tables, vous devez créer la base de données et la structure de ses tables. Ces tâches de configuration initiales moins fréquentes sont traitées à la section « Création et modification d'une base de données » à la page 746.

Vous pouvez choisir d'effectuer des opérations de base de données de façon asynchrone, ce qui signifie que le moteur de base de données s'exécute en arrière-plan et vous avertit en distribuant un événement lorsque l'opération réussit ou échoue. Vous pouvez également effectuer ces opérations de façon synchrone. Dans ce cas, les opérations de base de données sont exécutées l'une après l'autre et l'ensemble de l'application (y compris l'actualisation de l'écran) attend la fin des opérations avant d'exécuter le reste du code. Pour plus d'informations sur l'utilisation des modes d'exécution asynchrone ou synchrone, voir la section « Utilisation des opérations de base de données synchrones et asynchrones » à la page 782.

Connexion à une base de données

Adobe AIR 1.0 et les versions ultérieures

Avant d'effectuer toute opération sur une base de données, commencez par ouvrir une connexion au fichier de cette base de données. Une occurrence de SQLConnection permet de représenter une connexion à une ou plusieurs bases de données. La première base de données connectée par une occurrence de SQLConnection est appelée base de données « principale ». Cette base de données est connectée par la méthode `open()` (en mode d'exécution synchrone) ou par la méthode `openAsync()` (en mode d'exécution asynchrone).

Si vous ouvrez une base de données via l'opération asynchrone `openAsync()`, enregistrez l'événement `open` de l'occurrence de SQLConnection pour être averti(e) lorsque l'opération `openAsync()` se termine. Enregistrez l'événement `error` de l'occurrence de SQLConnection pour savoir si l'opération a échoué.

L'exemple suivant montre comment ouvrir un fichier de base de données existant pour une exécution asynchrone. Le fichier de base de données est nommé « DBSample.db » et réside dans le « Pointage vers le répertoire de stockage d'une application » à la page 697.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

conn.openAsync(dbFile, SQLMode.UPDATE);

function openHandler(event:SQLEvent):void
{
    trace("the database opened successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLMode;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                conn.openAsync(dbFile, SQLMode.UPDATE);
            }

            private function openHandler(event:SQLEvent):void
            {
                trace("the database opened successfully");
            }

            private function errorHandler(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

L'exemple suivant illustre l'ouverture d'un fichier de base de données existant pour une exécution synchrone. Le fichier de base de données est nommé « DBSample.db » et réside dans le « Pointage vers le répertoire de stockage d'une application » à la page 697.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.errors.SQLError;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

try
{
    conn.open(dbFile, SQLMode.UPDATE);
    trace("the database opened successfully");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLMode;
            import flash.errors.SQLError;
            import flash.filesystem.File;

            private function init():void
            {
                var conn:SQLConnection = new SQLConnection();

                // The database file is in the application storage directory
                var folder:File = File.applicationStorageDirectory;
                var dbFile:File = folder.resolvePath("DBSample.db");

                try
                {
                    conn.open(dbFile, SQLMode.UPDATE);
                    trace("the database opened successfully");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Notez que dans l'appel de méthode `openAsync()` de l'exemple asynchrone et dans l'appel de méthode `open()` de l'exemple synchrone, le second argument est la constante `SQLMode.UPDATE`. La définition de `SQLMode.UPDATE` pour le second paramètre (`openMode`) oblige le moteur d'exécution à distribuer une erreur si le fichier spécifié n'existe pas. Si vous transmettez `SQLMode.CREATE` pour le paramètre `openMode` (ou si vous laissez le paramètre `openMode` désactivé), le moteur d'exécution tente de créer un fichier de base de données lorsque le fichier spécifié n'existe pas. Toutefois, s'il existe, le fichier est ouvert, ce qui revient à utiliser `SQLMode.Update`. Vous pouvez également spécifier `SQLMode.READ` pour le paramètre `openMode` afin d'ouvrir une base de données existante en lecture seule. Dans ce cas, les données peuvent être récupérées dans la base de données, mais ne peuvent pas être ajoutées, supprimées ou modifiées.

Utilisation des instructions SQL

Adobe AIR 1.0 et les versions ultérieures

Une instruction SQL individuelle (requête ou commande) est représentée dans le moteur d'exécution par un objet SQLStatement. Pour créer et exécuter une instruction SQL, procédez comme suit :

Créez une occurrence de SQLStatement.
L'objet SQLStatement représente l'instruction SQL dans votre application.

```
var selectData:SQLStatement = new SQLStatement();
```

Spécifiez sur quelle base de données la requête doit s'exécuter.
Pour ce faire, définissez la propriété `sqlConnection` de l'objet SQLStatement sur l'occurrence de SQLConnection connectée à la base de données désirée.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

Définissez la véritable instruction SQL.
Créez le texte de l'instruction sous forme d'occurrence de String et affectez-la à la propriété `text` de l'occurrence de SQLStatement.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

Définissez les fonctions gérant le résultat de l'exécution (mode d'exécution asynchrone uniquement).
Utilisez la méthode `addEventListener()` pour enregistrer les fonctions en tant qu'écouteurs des événements `result` et `error` de l'occurrence de SQLStatement.

```
// using listener methods and addEventListener()

selectData.addEventListener(SQLEvent.RESULT, resultHandler);
selectData.addEventListener(SQLErrorEvent.ERROR, errorHandler);

function resultHandler(event:SQLEvent):void
{
    // do something after the statement execution succeeds
}

function errorHandler(event:SQLErrorEvent):void
{
    // do something after the statement execution fails
}
```

Vous pouvez également spécifier des méthodes d'écouteur d'événements à l'aide d'un objet Responder. Dans ce cas, créez l'occurrence de Responder et associez-la aux méthodes d'écouteur d'événement.

```
// using a Responder (flash.net.Responder)

var selectResponder = new Responder(onResult, onError);

function onResult(result:SQLResult):void
{
    // do something after the statement execution succeeds
}

function onError(error:SQLError):void
{
    // do something after the statement execution fails
}
```

Si le texte de l'instruction comprend des définitions de paramètres, donnez les valeurs de ces paramètres.
Pour affecter les valeurs des paramètres, utilisez la propriété de tableau associatif `parameters` de l'occurrence de SQLStatement.

```
selectData.parameters[":param1"] = 25;
```

Exécutez l'instruction SQL.
Appelez la méthode `execute()` de l'occurrence de SQLStatement.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

En outre, si vous utilisez une occurrence de Responder à la place des écouteurs d'événement en mode d'exécution asynchrone, transmettez cette occurrence à la méthode `execute()`.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

Vous trouverez des exemples spécifiques de ces procédures aux rubriques suivantes :

« Récupération de données dans une base de données » à la page 761

« Insertion de données » à la page 771

« Modification ou suppression de données » à la page 777

Utilisation de paramètres dans des instructions

Adobe AIR 1.0 et les versions ultérieures

L'ajout d'un paramètre dans une instruction SQL permet de créer une instruction SQL réutilisable. Lorsque vous ajoutez des paramètres à une instruction, les valeurs de celle-ci peuvent changer (les valeurs ajoutées à une instruction `INSERT`, par exemple), mais le texte de base de l'instruction ne change pas. C'est pourquoi l'ajout de paramètres constitue un avantage en termes de performances et simplifie le codage des applications.

Présentation des paramètres d'instruction

Adobe AIR 1.0 et les versions ultérieures

Il est fréquent qu'une application utilise plusieurs fois une même instruction SQL, avec de légères variations. Prenons par exemple le cas d'une application de suivi de stock qui permet à l'utilisateur d'ajouter de nouveaux articles dans la base de données. Le code de l'application qui ajoute un article de stock dans la base de données exécute une instruction SQL `INSERT` qui ajoute véritablement les données dans la base de données. Toutefois, chaque exécution de l'instruction présente une légère variation. En particulier, les véritables valeurs insérées dans la table diffèrent puisqu'elles sont spécifiques à l'article ajouté au stock.

Lorsqu'une instruction SQL est utilisée plusieurs fois avec des valeurs différentes dans l'instruction, la meilleure approche consiste à utiliser une instruction SQL incluant des paramètres plutôt que des valeurs littérales dans le texte SQL. Un paramètre est un espace réservé dans le texte de l'instruction qui est remplacé par une valeur réelle à chaque exécution de l'instruction. Pour utiliser des paramètres dans une instruction SQL, vous créez une occurrence de SQLStatement standard. Dans le cas de la véritable instruction SQL affectée à la propriété `text`, utilisez des espaces réservés aux paramètres plutôt que des valeurs littérales. Définissez ensuite la valeur de chaque paramètre en définissant la valeur d'un élément dans la propriété `parameters` de l'occurrence de SQLStatement. La propriété `parameters` étant un tableau associatif, définissez une valeur déterminée à l'aide de la syntaxe suivante :

```
statement.parameters[parameter_identifier] = value;
```

parameter_identifier est une chaîne si vous utilisez un paramètre nommé ou un index de nombres entiers si vous utilisez un paramètre non nommé.

Utilisation de paramètres nommés

Adobe AIR 1.0 et les versions ultérieures

Un paramètre peut être un paramètre nommé. Un paramètre nommé a un nom spécifique que la base de données utilise pour mettre en correspondance sa valeur avec l'emplacement de l'espace réservé dans le texte de l'instruction. Un nom de paramètre se compose du caractère « : » ou « @ » suivi d'un nom, comme dans les exemples suivants :

```
:itemName
@firstName
```

Le code suivant décrit l'utilisation des paramètres nommés :

```
var sql:String =
    "INSERT INTO inventoryItems (name, productCode)" +
    "VALUES (:name, :productCode)";

var addItemStmt:SQLStatement = new SQLStatement();
addItemStmt.sqlConnection = conn;
addItemStmt.text = sql;

// set parameter values
addItemStmt.parameters[":name"] = "Item name";
addItemStmt.parameters[":productCode"] = "12345";

addItemStmt.execute();
```

Utilisation de paramètres non nommés

Adobe AIR 1.0 et les versions ultérieures

En alternative à l'utilisation de paramètres nommés, vous pouvez utiliser des paramètres non nommés. Pour utiliser un paramètre non nommé, vous désignez un paramètre dans une instruction SQL en utilisant un caractère « ? » . Un index numérique est affecté à chaque paramètre, par ordre d'apparition des paramètres dans l'instruction, en commençant par l'index 0 pour le premier paramètre. L'exemple suivant est une version différente de l'exemple précédent, à l'aide de paramètres non nommés :

```
var sql:String =
    "INSERT INTO inventoryItems (name, productCode)" +
    "VALUES (?, ?)";

var addItemStmt:SQLStatement = new SQLStatement();
addItemStmt.sqlConnection = conn;
addItemStmt.text = sql;

// set parameter values
addItemStmt.parameters[0] = "Item name";
addItemStmt.parameters[1] = "12345";

addItemStmt.execute();
```

Avantages de l'utilisation de paramètres

Adobe AIR 1.0 et les versions ultérieures

L'utilisation de paramètres dans une instruction SQL présente plusieurs avantages :

Performances optimisées L'exécution d'une occurrence de SQLStatement avec paramètres est plus efficace que celle d'une occurrence qui crée dynamiquement le texte SQL à chaque exécution. L'amélioration des performances est due au fait que l'instruction n'est préparée qu'une seule fois mais peut ensuite être exécutée à plusieurs reprises avec des valeurs différentes de paramètres, sans qu'il soit nécessaire de recompiler l'instruction SQL.

Typage explicite des données Les paramètres autorisent la substitution avec type de valeurs inconnues au moment de la construction de l'instruction SQL. L'utilisation des paramètres est le seul moyen de garantir la classe de stockage d'une valeur transmise à la base de données. Lorsque les paramètres ne sont pas utilisés, le moteur d'exécution tente de convertir toutes les valeurs de leur représentation texte en une classe de stockage en fonction de l'affinité du type de la colonne associée.

Pour plus d'informations sur les classes de stockage et l'affinité des colonnes, voir « Prise en charge des types de données » à la page 1177.

Sécurité renforcée Les paramètres sont également utilisés comme mesure de sécurité pour prévenir toute technique malveillante appelée attaque par injection de code SQL. Dans une attaque par injection de code SQL, l'utilisateur entre du code SQL dans un emplacement accessible (par exemple dans un champ de saisie). Si le code de l'application construit une instruction SQL en concaténant directement la saisie de l'utilisateur dans le texte SQL, le code SQL saisi par l'utilisateur est exécuté sur la base de données. L'exemple suivant illustre la concaténation de la saisie de l'utilisateur dans le texte SQL. **N'utilisez pas cette technique** :

```
// assume the variables "username" and "password"
// contain user-entered data

var sql:String =
    "SELECT userId " +
    "FROM users " +
    "WHERE username = '" + username + "' " +
    "    AND password = '" + password + "'";

var statement:SQLStatement = new SQLStatement();
statement.text = sql;
```

Le fait d'utiliser des paramètres d'instruction plutôt que la concaténation de valeurs saisies par utilisateur dans le texte d'une instruction permet d'éviter les attaques par injection de code SQL. L'attaque par injection de code SQL ne peut se produire car les valeurs des paramètres sont traitées explicitement sous forme de valeurs substituées au lieu de devenir une partie du texte de l'instruction littérale. L'exemple suivant est l'alternative recommandée :

```
// assume the variables "username" and "password"
// contain user-entered data

var sql:String =
    "SELECT userId " +
    "FROM users " +
    "WHERE username = :username " +
    "    AND password = :password";

var statement:SQLStatement = new SQLStatement();
statement.text = sql;

// set parameter values
statement.parameters[":username"] = username;
statement.parameters[":password"] = password;
```

Récupération de données dans une base de données

Adobe AIR 1.0 et les versions ultérieures

La récupération de données dans une base de données comprend deux étapes. Vous exécutez d'abord une instruction SQL `SELECT`, en décrivant le jeu de données désiré de la base de données. Vous accédez ensuite aux données récupérées et vous les affichez ou les manipulez selon les besoins de votre application.

Exécution d'une instruction SELECT

Adobe AIR 1.0 et les versions ultérieures

Pour extraire des données existantes d'une base de données, utilisez une occurrence de SQLStatement. Affectez l'instruction SQL `SELECT` appropriée à la propriété `text` de l'occurrence, puis appelez sa méthode `execute()`.

Pour plus d'informations sur la syntaxe de l'instruction `SELECT`, voir « Prise en charge de SQL dans les bases de données locales » à la page 1154.

L'exemple suivant décrit l'exécution d'une instruction `SELECT` pour récupérer des données dans une table nommée « products », en mode d'exécution asynchrone :

```
var selectStmt:SQLStatement = new SQLStatement();

// A SQLConnection named "conn" has been created previously
selectStmt.sqlConnection = conn;

selectStmt.text = "SELECT itemId, itemName, price FROM products";

selectStmt.addEventListener(SQLEvent.RESULT, resultHandler);
selectStmt.addEventListener(SQLErrorEvent.ERROR, errorHandler);

selectStmt.execute();

function resultHandler(event:SQLEvent):void
{
    var result:SQLResult = selectStmt.getResult();

    var numResults:int = result.data.length;
    for (var i:int = 0; i < numResults; i++)
    {
        var row:Object = result.data[i];
        var output:String = "itemId: " + row.itemId;
        output += "; itemName: " + row.itemName;
        output += "; price: " + row.price;
        trace(output);
    }
}

function errorHandler(event:SQLErrorEvent):void
{
    // Information about the error is available in the
    // event.error property, which is an instance of
    // the SQLError class.
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                var selectStmt:SQLStatement = new SQLStatement();

                // A SQLConnection named "conn" has been created previously
                selectStmt.sqlConnection = conn;

                selectStmt.text = "SELECT itemId, itemName, price FROM products";

                selectStmt.addEventListener(SQLEvent.RESULT, resultHandler);
                selectStmt.addEventListener(SQLErrorEvent.ERROR, errorHandler);
```

```
            selectStmt.execute();
        }

        private function resultHandler(event:SQLEvent):void
        {
            var result:SQLResult = selectStmt.getResult();

            var numResults:int = result.data.length;
            for (var i:int = 0; i < numResults; i++)
            {
                var row:Object = result.data[i];
                var output:String = "itemId: " + row.itemId;
                output += "; itemName: " + row.itemName;
                output += "; price: " + row.price;
                trace(output);
            }
        }

        private function errorHandler(event:SQLErrorEvent):void
        {
            // Information about the error is available in the
            // event.error property, which is an instance of
            // the SQLError class.
        }
    ]]>
</mx:Script>
</mx:WindowedApplication>
```

L'exemple suivant décrit l'exécution d'une instruction `SELECT` pour récupérer des données dans une table nommée « products », en mode d'exécution synchrone :

```
var selectStmt:SQLStatement = new SQLStatement();

// A SQLConnection named "conn" has been created previously
selectStmt.sqlConnection = conn;

selectStmt.text = "SELECT itemId, itemName, price FROM products";

try
{
    selectStmt.execute();

    var result:SQLResult = selectStmt.getResult();

    var numResults:int = result.data.length;
    for (var i:int = 0; i < numResults; i++)
    {
        var row:Object = result.data[i];
        var output:String = "itemId: " + row.itemId;
        output += "; itemName: " + row.itemName;
        output += "; price: " + row.price;
        trace(output);
    }
}
catch (error:SQLError)
{
    // Information about the error is available in the
    // error variable, which is an instance of
    // the SQLError class.
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                var selectStmt:SQLStatement = new SQLStatement();

                // A SQLConnection named "conn" has been created previously
                selectStmt.sqlConnection = conn;

                selectStmt.text = "SELECT itemId, itemName, price FROM products";

                try
                {
                    selectStmt.execute();

                    var result:SQLResult = selectStmt.getResult();
```

```
                    var numResults:int = result.data.length;
                    for (var i:int = 0; i < numResults; i++)
                    {
                        var row:Object = result.data[i];
                        var output:String = "itemId: " + row.itemId;
                        output += "; itemName: " + row.itemName;
                        output += "; price: " + row.price;
                        trace(output);
                    }
                }
                catch (error:SQLError)
                {
                    // Information about the error is available in the
                    // error variable, which is an instance of
                    // the SQLError class.
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

En mode d'exécution asynchrone, lorsque l'instruction se termine, l'occurrence de SQLStatement déclenche un événement `result` (`SQLEvent.RESULT`) indiquant que l'instruction s'est exécutée avec succès. Sinon, lorsqu'un objet Responder est transmis en tant qu'argument à la méthode `execute()`, la fonction du gestionnaire de résultats de l'objet Responder est appelée. En mode d'exécution synchrone, l'exécution s'interrompt jusqu'à la fin de l'opération `execute()`, puis passe à la ligne de code suivante.

Accès aux données du résultat de l'instruction SELECT

Adobe AIR 1.0 et les versions ultérieures

Lorsque l'exécution de l'instruction `SELECT` est terminée, l'étape suivante consiste à accéder aux données récupérées. Pour récupérer les données du résultat de l'instruction `SELECT` exécutée, appelez la méthode `getResult()` de l'objet SQLStatement :

```
var result:SQLResult = selectStatement.getResult();
```

La méthode `getResult()` renvoie un objet SQLResult. La propriété `data` de l'objet SQLResult est un tableau qui contient les résultats de l'instruction `SELECT` :

```
var numResults:int = result.data.length;
for (var i:int = 0; i < numResults; i++)
{
    // row is an Object representing one row of result data
    var row:Object = result.data[i];
}
```

Chaque ligne de données du jeu de résultats de l'instruction `SELECT` devient une occurrence de l'objet dans le tableau `data`. Cet objet a les propriétés dont les noms correspondent aux noms des colonnes du jeu de résultats. Les propriétés contiennent les valeurs provenant des colonnes du jeu de résultats. Par exemple, supposons qu'une instruction `SELECT` spécifie un jeu de résultats avec trois colonnes nommées « itemId », « itemName » et « price ». Pour chaque ligne du jeu de résultats, une occurrence d'Object est créée avec les propriétés nommées `itemId`, `itemName` et `price`. Ces propriétés contiennent les valeurs de leurs colonnes respectives.

Le code suivant définit une occurrence de SQLStatement dont le texte est une instruction `SELECT`. L'instruction récupère les lignes contenant les valeurs des colonnes `firstName` et `lastName` de toutes les lignes d'une table nommée `employees`. Cet exemple utilise le mode d'exécution asynchrone. Lorsque l'exécution est terminée, la méthode `selectResult()` est appelée, et vous accédez aux lignes de données résultantes à l'aide de la méthode `SQLStatement.getResult()` et vous les affichez à l'aide de la méthode `trace()`. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à la base de données. Il suppose également que la table « employees » a déjà été créée et contient des données.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var selectStmt:SQLStatement = new SQLStatement();
selectStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "SELECT firstName, lastName " +
    "FROM employees";
selectStmt.text = sql;

// register listeners for the result and error events
selectStmt.addEventListener(SQLEvent.RESULT, selectResult);
selectStmt.addEventListener(SQLErrorEvent.ERROR, selectError);

// execute the statement
selectStmt.execute();

function selectResult(event:SQLEvent):void
{
    // access the result data
    var result:SQLResult = selectStmt.getResult();

    var numRows:int = result.data.length;
    for (var i:int = 0; i < numRows; i++)
    {
        var output:String = "";
        for (var columnName:String in result.data[i])
        {
            output += columnName + ": " + result.data[i][columnName] + "; ";
        }
        trace("row[" + i.toString() + "]\t", output);
    }
}

function selectError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var selectStmt:SQLStatement = new SQLStatement();
                selectStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "SELECT firstName, lastName " +
                    "FROM employees";
                selectStmt.text = sql;

                // register listeners for the result and error events
                selectStmt.addEventListener(SQLEvent.RESULT, selectResult);
                selectStmt.addEventListener(SQLErrorEvent.ERROR, selectError);

                // execute the statement
                selectStmt.execute();
            }

            private function selectResult(event:SQLEvent):void
            {
                // access the result data
                var result:SQLResult = selectStmt.getResult();

                var numRows:int = result.data.length;
                for (var i:int = 0; i < numRows; i++)
                {
                    var output:String = "";
                    for (var columnName:String in result.data[i])
                    {
                        output += columnName + ": " + result.data[i][columnName] + "; ";
                    }
                    trace("row[" + i.toString() + "]\t", output);
                }
            }

            private function selectError(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Le code suivant décrit les mêmes techniques que le précédent, mais avec le mode d'exécution synchrone. L'exemple définit une occurrence de SQLStatement dont le texte est une instruction `SELECT`. L'instruction récupère les lignes contenant les valeurs des colonnes `firstName` et `lastName` de toutes les lignes d'une table nommée `employees`. Vous accédez aux lignes de données résultantes à l'aide de la méthode `SQLStatement.getResult()` et vous les affichez à l'aide de la méthode `trace()`. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à la base de données. Il suppose également que la table « employees » a déjà été créée et contient des données.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var selectStmt:SQLStatement = new SQLStatement();
selectStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "SELECT firstName, lastName " +
    "FROM employees";
selectStmt.text = sql;

try
{
    // execute the statement
    selectStmt.execute();

    // access the result data
    var result:SQLResult = selectStmt.getResult();

    var numRows:int = result.data.length;
    for (var i:int = 0; i < numRows; i++)
    {
        var output:String = "";
        for (var columnName:String in result.data[i])
        {
            output += columnName + ": " + result.data[i][columnName] + "; ";
        }
        trace("row[" + i.toString() + "]\t", output);
    }
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var selectStmt:SQLStatement = new SQLStatement();
                selectStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "SELECT firstName, lastName " +
                    "FROM employees";
                selectStmt.text = sql;

                try
                {
                    // execute the statement
                    selectStmt.execute();

                    // access the result data
                    var result:SQLResult = selectStmt.getResult();

                    var numRows:int = result.data.length;
                    for (var i:int = 0; i < numRows; i++)
                    {
                        var output:String = "";
                        for (var columnName:String in result.data[i])
                        {
                            output += columnName + ": ";
                            output += result.data[i][columnName] + "; ";
                        }
                        trace("row[" + i.toString() + "]\t", output);
                    }
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Définition du type des données du résultat de l'instruction SELECT

Adobe AIR 1.0 et les versions ultérieures

Par défaut, chaque ligne renvoyée par une instruction `SELECT` est créée en tant qu'occurrence d'objet dont les noms de propriétés correspondent aux noms de colonnes du jeu de résultats et la valeur de chaque colonne à la valeur de la propriété associée. Toutefois, avant d'exécuter une instruction SQL `SELECT`, vous pouvez définir la propriété `itemClass` de l'occurrence de SQLStatement sur une classe. En définissant la propriété `itemClass`, chaque ligne renvoyée par l'instruction `SELECT` est créée sous la forme d'une occurrence de la classe désignée. Le moteur d'exécution affecte les valeurs des propriétés aux valeurs des colonnes de résultats en mettant en correspondance les noms de colonnes du jeu de résultats `SELECT` et les noms des propriétés de la classe `itemClass`.

Toute classe affectée en tant que propriété `itemClass` doit avoir un constructeur qui ne requiert aucun paramètre. En outre, la classe doit avoir une seule propriété pour chaque colonne renvoyée par l'instruction `SELECT`. Le fait qu'une colonne de la liste `SELECT` ne présente pas de nom de propriété correspondant dans la classe `itemClass` est considéré comme une erreur.

Récupération partielle des résultats d'une instruction SELECT

Adobe AIR 1.0 et les versions ultérieures

Par défaut, l'exécution d'une instruction `SELECT` récupère simultanément toutes les lignes du jeu de résultats. Une fois l'instruction terminée, vous traitez généralement les données récupérées d'une manière ou d'une autre, par exemple en créant des objets ou en affichant les données à l'écran. Si l'instruction renvoie un très grand nombre de lignes, le traitement simultané de toutes les données peut se révéler très exigeant pour l'ordinateur, qui à son tour peut ne pas redessiner l'interface pour l'utilisateur.

Pour améliorer les performances de votre application, vous pouvez demander au moteur d'exécution de ne renvoyer simultanément qu'un nombre spécifique de lignes de résultats. Les données de résultats initiales sont ainsi renvoyées plus rapidement. Vous pouvez également diviser les lignes du résultat en jeux, de sorte que l'interface utilisateur soit mise à jour après le traitement de chaque jeu de lignes. Notez que cette technique n'est pratique qu'en mode d'exécution asynchrone.

Pour récupérer les résultats partiels de `SELECT`, donnez une valeur au premier paramètre de la méthode `SQLStatement.execute()` (paramètre `prefetch`). Le paramètre `prefetch` indique le nombre de lignes à récupérer à la première exécution de l'instruction. Lorsque vous appelez la méthode `execute()` d'une occurrence de SQLStatement, spécifiez la valeur du paramètre `prefetch` afin que seul ce nombre de lignes soit récupéré :

```
var stmt:SQLStatement = new SQLStatement();
stmt.sqlConnection = conn;

stmt.text = "SELECT ...";

stmt.addEventListener(SQLEvent.RESULT, selectResult);

stmt.execute(20); // only the first 20 rows (or fewer) are returned
```

L'instruction déclenche l'événement `result`, qui indique que le premier jeu de lignes de résultat est disponible. La propriété `data` de l'occurrence de SQLResult résultante contient les lignes de données, et sa propriété `complete` indique s'il reste d'autres lignes de résultat à récupérer. Pour récupérer ces autres lignes, appelez la méthode `next()` de l'occurrence de SQLStatement. Comme la méthode `execute()`, le premier paramètre de la méthode `next()` sert à indiquer le nombre de lignes à récupérer lors du prochain déclenchement de l'événement de résultat.

```
function selectResult(event:SQLEvent):void
{
    var result:SQLResult = stmt.getResult();
    if (result.data != null)
    {
        // ... loop through the rows or perform other processing ...

        if (!result.complete)
        {
            stmt.next(20); // retrieve the next 20 rows
        }
        else
        {
            stmt.removeEventListener(SQLEvent.RESULT, selectResult);
        }
    }
}
```

L'occurrence de SQLStatement déclenche un événement `result` chaque fois que la méthode `next()` renvoie un jeu suivant de lignes de résultat. Par conséquent, la même fonction d'écouteur peut être utilisée pour continuer à traiter les résultats (par des appels à `next()`) jusqu'à ce que toutes les lignes aient été récupérées.

Pour plus d'informations, voir les descriptions des méthodes `SQLStatement.execute()` (description du paramètre `prefetch`) et `SQLStatement.next()`.

Insertion de données

Adobe AIR 1.0 et les versions ultérieures

L'ajout de données dans une base de données implique l'exécution d'une instruction SQL `INSERT`. Lorsque l'exécution de l'instruction est terminée, vous pouvez accéder à la clé primaire de la ligne nouvellement insérée si la base de données en a généré une.

Exécution d'une instruction INSERT

Adobe AIR 1.0 et les versions ultérieures

Pour ajouter des données dans une table de base de données, créez et exécutez une occurrence de SQLStatement dont le texte est une instruction SQL `INSERT`.

L'exemple suivant utilise une occurrence de SQLStatement pour ajouter une ligne de données dans la table des employés déjà existante. Cet exemple décrit l'insertion de données en mode d'exécution asynchrone. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à une base de données. Il suppose également que la table « employees » a déjà été créée.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var insertStmt:SQLStatement = new SQLStatement();
insertStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "INSERT INTO employees (firstName, lastName, salary) " +
    "VALUES ('Bob', 'Smith', 8000)";
insertStmt.text = sql;

// register listeners for the result and failure (status) events
insertStmt.addEventListener(SQLEvent.RESULT, insertResult);
insertStmt.addEventListener(SQLErrorEvent.ERROR, insertError);

// execute the statement
insertStmt.execute();

function insertResult(event:SQLEvent):void
{
    trace("INSERT statement succeeded");
}

function insertError(event:SQLErrorEvent):void
{
    trace("Error message:", event.error.message);
    trace("Details:", event.error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.events.SQLErrorEvent;
            import flash.events.SQLEvent;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var insertStmt:SQLStatement = new SQLStatement();
                insertStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "INSERT INTO employees (firstName, lastName, salary) " +
```

```
                    "VALUES ('Bob', 'Smith', 8000)";
                insertStmt.text = sql;

                // register listeners for the result and failure (status) events
                insertStmt.addEventListener(SQLEvent.RESULT, insertResult);
                insertStmt.addEventListener(SQLErrorEvent.ERROR, insertError);

                // execute the statement
                insertStmt.execute();
            }

            private function insertResult(event:SQLEvent):void
            {
                trace("INSERT statement succeeded");
            }

            private function insertError(event:SQLErrorEvent):void
            {
                trace("Error message:", event.error.message);
                trace("Details:", event.error.details);
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

L'exemple suivant ajoute une ligne de données à la table des employés existante, en mode d'exécution synchrone. Notez que ce code présuppose l'existence d'une occurrence de SQLConnection nommée `conn`, déjà instanciée et connectée à une base de données. Il suppose également que la table « employees » a déjà été créée.

```
import flash.data.SQLConnection;
import flash.data.SQLResult;
import flash.data.SQLStatement;
import flash.errors.SQLError;

// ... create and open the SQLConnection instance named conn ...

// create the SQL statement
var insertStmt:SQLStatement = new SQLStatement();
insertStmt.sqlConnection = conn;

// define the SQL text
var sql:String =
    "INSERT INTO employees (firstName, lastName, salary) " +
    "VALUES ('Bob', 'Smith', 8000)";
insertStmt.text = sql;

try
{
    // execute the statement
    insertStmt.execute();

    trace("INSERT statement succeeded");
}
catch (error:SQLError)
{
    trace("Error message:", error.message);
    trace("Details:", error.details);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.data.SQLConnection;
            import flash.data.SQLResult;
            import flash.data.SQLStatement;
            import flash.errors.SQLError;

            private function init():void
            {
                // ... create and open the SQLConnection instance named conn ...

                // create the SQL statement
                var insertStmt:SQLStatement = new SQLStatement();
                insertStmt.sqlConnection = conn;

                // define the SQL text
                var sql:String =
                    "INSERT INTO employees (firstName, lastName, salary) " +
                    "VALUES ('Bob', 'Smith', 8000)";
                insertStmt.text = sql;

                try
                {
                    // execute the statement
                    insertStmt.execute();
                    trace("INSERT statement succeeded");
                }
                catch (error:SQLError)
                {
                    trace("Error message:", error.message);
                    trace("Details:", error.details);
                }
            }
        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

Récupération de la clé primaire d'une ligne insérée, générée par la base de données

Adobe AIR 1.0 et les versions ultérieures

Après l'insertion d'une ligne de données dans une table, il arrive souvent que le code doive connaître la clé primaire générée par la base de données ou une valeur d'identificateur de ligne pour la ligne nouvellement insérée. Par exemple, après avoir inséré une ligne dans une table, vous pouvez en ajouter d'autres dans une table associée. Dans ce cas, vous pouvez insérer la valeur de la clé primaire en tant que clé étrangère de la table associée. La clé primaire d'une ligne nouvellement insérée peut être récupérée par le biais de l'objet SQLResult généré par l'instruction exécutée. Il s'agit du même objet qui est utilisé pour accéder aux données du résultat après l'exécution d'une instruction `SELECT`. Comme pour toute instruction SQL, lorsque l'exécution d'une instruction `INSERT` se termine, le moteur d'exécution crée une occurrence de SQLResult. Pour accéder à l'occurrence de SQLResult, vous appelez la méthode `getResult()` de l'objet

SQLStatement si vous utilisez un écouteur d'événements ou le mode d'exécution synchrone. Si vous utilisez le mode d'exécution asynchrone et que vous transmettez une occurrence de Responder à l'appel de la méthode `execute()`, l'occurrence de SQLResult est transmise en tant qu'argument à la fonction du gestionnaire de résultat. Dans tous les cas, l'occurrence de SQLResult a une propriété, `lastInsertRowID`, qui contient l'identificateur de la dernière ligne insérée si l'instruction SQL exécutée est une instruction `INSERT`.

L'exemple suivant montre comment accéder à la clé primaire d'une ligne insérée en mode d'exécution asynchrone :

```
insertStmt.text = "INSERT INTO ...";

insertStmt.addEventListener(SQLEvent.RESULT, resultHandler);

insertStmt.execute();

function resultHandler(event:SQLEvent):void
{
    // get the primary key
    var result:SQLResult = insertStmt.getResult();

    var primaryKey:Number = result.lastInsertRowID;
    // do something with the primary key
}
```

L'exemple suivant montre comment accéder à la clé primaire d'une ligne insérée en mode d'exécution synchrone :

```
insertStmt.text = "INSERT INTO ...";

try
{
    insertStmt.execute();

    // get the primary key
    var result:SQLResult = insertStmt.getResult();

    var primaryKey:Number = result.lastInsertRowID;
    // do something with the primary key
}
catch (error:SQLError)
{
    // respond to the error
}
```

Notez que l'identificateur de ligne peut ou non correspondre à la valeur de la colonne désignée en tant que colonne de clé primaire dans la définition de la table, en fonction des règles suivantes :

- Si la table est définie avec une colonne de clé primaire dont l'affinité (son type de données) est `INTEGER`, la propriété `lastInsertRowID` contient la valeur insérée dans cette ligne (ou celle générée par le moteur d'exécution s'il s'agit d'une colonne `AUTOINCREMENT`).
- Si la table comporte plusieurs colonnes de clés primaires (clé composite) ou une seule colonne de clés primaires dont l'affinité n'est pas `INTEGER`, la base de données génère une valeur d'identificateur de ligne entier en arrière-plan. La valeur générée est celle de la propriété `lastInsertRowID`.
- La valeur est toujours l'identificateur de la dernière ligne insérée. Si une instruction `INSERT` provoque un déclencheur qui à son tour insère une ligne, la propriété `lastInsertRowID` contient l'identificateur de la dernière ligne insérée par le déclencheur et non la ligne créée par l'instruction `INSERT`.

La conséquence de ces règles est la suivante : pour obtenir une colonne de clés primaires définie de façon explicite dont la valeur est générée par une commande `INSERT` via la propriété `SQLResult.lastInsertRowID`, définissez la colonne en tant que colonne `INTEGER PRIMARY KEY`. Même si la table ne comprend pas de colonne `INTEGER PRIMARY KEY` explicite, il est également possible d'utiliser l'identificateur de ligne généré par la base de données comme clé primaire de la table (définition des relations avec les tables associées). La valeur de colonne de l'identificateur de ligne est disponible dans toute instruction SQL via l'utilisation des noms de colonne spéciaux `ROWID`, `_ROWID_` ou `OID`. Vous pouvez créer une colonne de clé étrangère dans une table associée et utiliser la valeur de l'identificateur de ligne en tant que valeur de colonne de clé étrangère comme vous le feriez dans le cas d'une colonne `INTEGER PRIMARY KEY` explicitement déclarée. Ainsi, si vous utilisez une clé primaire arbitraire plutôt qu'une clé naturelle, et tant que la génération par le moteur d'exécution d'une valeur de clé primaire à votre place ne vous dérange pas, l'utilisation d'une colonne `INTEGER PRIMARY KEY` ou d'un identificateur de ligne généré par le système comme clé primaire de la table ne présente que peu de différence quant à la définition d'une relation de clé étrangère entre les deux tables.

Pour plus d'informations sur les clés principales et les identificateurs de ligne générés, voir « Prise en charge de SQL dans les bases de données locales » à la page 1154.

Modification ou suppression de données

Adobe AIR 1.0 et les versions ultérieures

La procédure d'exécution des autres opérations de manipulation des données est identique à celle utilisée pour exécuter une instruction SQL `SELECT` ou `INSERT` (voir « Utilisation des instructions SQL » à la page 757). Substituez simplement une autre instruction SQL dans la propriété `text` de l'occurrence de SQLStatement :

- Pour modifier les données existantes d'une table, utilisez une instruction `UPDATE`.
- Pour supprimer une ou plusieurs lignes de données dans une table, utilisez une instruction `DELETE`.

Pour consulter une description de ces instructions, voir « Prise en charge de SQL dans les bases de données locales » à la page 1154.

Utilisation de plusieurs bases de données

Adobe AIR 1.0 et les versions ultérieures

La méthode `SQLConnection.attach()` permet d'ouvrir une connexion à une autre base de données sur une occurrence de SQLConnection pour laquelle une base de données est déjà ouverte. Nommez la base de données reliée à l'aide du paramètre name dans l'appel de la méthode `attach()`. Lorsque vous écrivez des instructions pour manipuler cette base de données, vous pouvez alors utiliser ce nom dans un préfixe (sous la forme `nom-base de données.nom-table`) pour qualifier les noms de table dans vos instructions SQL, indiquant ainsi au moteur d'exécution que la table est disponible dans la base de données nommée.

Vous pouvez exécuter une même instruction SQL incluant des tables de plusieurs bases de données connectées à la même occurrence de SQLConnection. Si une transaction est créée sur l'occurrence de SQLConnection, cette transaction s'applique à toutes les instructions SQL exécutées à l'aide de cette occurrence de SQLConnection. Cela est vrai quelle que soit la base de données reliée sur laquelle l'instruction s'exécute.

Vous pouvez également créer plusieurs occurrences de SQLConnection dans une application, chacune connectée à une ou plusieurs bases de données. Toutefois, si vous utilisez cette technique, n'oubliez pas qu'une transaction de bases de données n'est pas partagée entre les occurrences de SQLConnection. Par conséquent, si vous vous connectez au même fichier de base de données à l'aide de plusieurs occurrences de SQLConnection, ne vous attendez pas à ce que les modifications de données des deux connexions s'appliquent de façon prévue. Par exemple, si deux instructions `UPDATE` ou `DELETE` s'exécutent sur la même base de données par l'intermédiaire d'occurrences de SQLConnection différentes et qu'une erreur survient après une opération, les données de la base peuvent être laissées dans un état intermédiaire éventuellement non réversible et affecter l'intégrité de la base de données (donc de l'application).

Gestion des erreurs de base de données

Adobe AIR 1.0 et les versions ultérieures

En général, la gestion des erreurs de base de données est similaire à celle des autres erreurs d'exécution. Il est préférable d'écrire du code préparé aux erreurs susceptibles de survenir, et de répondre aux erreurs plutôt que de laisser ce rôle au moteur d'exécution. De façon générale, les erreurs de base de données potentielles se divisent en trois catégories : les erreurs de connexion, les erreurs de syntaxe SQL et les erreurs de contrainte.

Erreurs de connexion

Adobe AIR 1.0 et les versions ultérieures

La plupart des erreurs de base de données sont des erreurs de connexion et peuvent survenir durant n'importe quelle opération. Si certaines stratégies permettent d'éviter les erreurs de connexion, il est rarement simple de récupérer correctement d'une erreur de connexion si la base de données est une partie sensible de votre application.

La plupart des erreurs de connexion sont liés aux interactions entre le moteur d'exécution et le système d'exploitation, le système de fichiers et le fichier de la base de données. Par exemple, une erreur de connexion se produit si l'utilisateur n'est pas autorisé à créer un fichier de base de données dans un emplacement particulier du système de fichiers. Les stratégies suivantes permettent d'éviter les erreurs de connexion :

Utilisation de fichiers de base de données spécifiques aux utilisateurs Au lieu d'utiliser un seul fichier de base de données pour tous les utilisateurs de l'application sur un même ordinateur, donnez à chaque utilisateur son propre fichier de base de données. Ce fichier doit être situé dans un répertoire associé au compte de l'utilisateur. Par exemple, il peut être placé dans le répertoire de stockage de l'application, le dossier Mes documents de l'utilisateur, sur le bureau de celui-ci, etc.

Prise en compte des différents types d'utilisateurs Testez votre application avec les différents types de comptes d'utilisateur, sur des systèmes d'exploitation différents. Ne supposez pas que l'utilisateur possède des privilèges d'administrateur sur l'ordinateur. De même, ne partez pas du principe que la personne qui a installé l'application est l'utilisateur qui l'exécute.

Prise en compte des divers emplacements des fichiers Si vous autorisez l'utilisateur à spécifier l'emplacement d'enregistrement d'un fichier de base de données ou à sélectionner le fichier à ouvrir, tenez compte des emplacements de fichiers auxquels les utilisateurs peuvent accéder. Pensez en outre à limiter les emplacements dans lesquels les utilisateurs peuvent stocker des fichiers de base de données (ou à partir desquels ils peuvent en ouvrir). Par exemple, vous pouvez autoriser uniquement les utilisateurs à ouvrir les fichiers situés dans l'emplacement de stockage de leur compte d'utilisateur.

Si une erreur de connexion se produit, elle surviendra probablement lors de la première tentative de création ou d'ouverture de la base de données. Cela signifie que l'utilisateur ne peut effectuer aucune opération de base de données dans l'application. Pour certains types d'erreurs, par exemple les erreurs d'autorisation ou de lecture seule, une technique de récupération possible consiste à copier le fichier de base de données dans un emplacement différent.

L'application peut copier les fichiers de bases de données dans un emplacement pour lequel l'utilisateur est autorisé à créer et à écrire dans des fichiers, et utiliser cet emplacement à la place.

Erreurs de syntaxe

Adobe AIR 1.0 et les versions ultérieures

Une erreur de syntaxe se produit lorsque l'application tente d'exécuter une instruction SQL qui n'est pas correctement rédigée. Les instructions SQL de base de données locales étant créées sous forme de chaîne, la vérification de la syntaxe SQL au moment de la compilation n'est pas possible. Toutes les instructions SQL doivent être exécutées pour vérifier leur syntaxe. Pour éviter les erreurs de syntaxe SQL, utilisez les stratégies suivantes :

Test approfondi de toutes les instructions SQL Si possible, testez vos instructions SQL séparément lors du développement de votre application avant de les coder sous forme de texte d'instruction dans le code de l'application. De plus, utilisez une approche de test de code telle que le test des unités pour créer un ensemble de tests vérifiant chaque option possible et chaque variation du code.

Utilisation des paramètres d'instruction au lieu de la concaténation (création dynamique) de code SQL Le fait d'utiliser des paramètres et d'éviter la construction dynamique d'instruction SQL permet d'utiliser le même texte d'instruction SQL à chaque exécution d'une instruction. En conséquence, le test de vos instructions est plus simple et les variations possibles sont limitées. Si vous devez générer une instruction SQL de façon dynamique, réduisez au strict minimum ses parties dynamiques. De même, validez soigneusement toutes les saisies éventuelles des utilisateurs afin de vous assurer qu'elles n'entraîneront pas d'erreur de syntaxe.

Pour récupérer d'une erreur de syntaxe, une application a besoin de code complexe pour pouvoir examiner l'instruction SQL et en corriger la syntaxe. En respectant les stratégies précédentes, votre code peut identifier toute source potentielle d'erreur de syntaxe SQL au moment de l'exécution (par exemple la saisie de l'utilisateur dans une instruction). Pour récupérer d'une erreur de syntaxe, donnez des consignes à l'utilisateur. Indiquez-lui ce qu'il doit faire pour que l'instruction s'exécute correctement.

Erreurs de contrainte

Adobe AIR 1.0 et les versions ultérieures

Les erreurs de contrainte se produisent lorsqu'une instruction `INSERT` ou `UPDATE` tente d'ajouter des données dans une colonne. L'erreur survient si les nouvelles données ne respectent pas l'une des contraintes définies pour la table ou la colonne. L'ensemble des contraintes possibles comprend :

Contrainte unique Indique que pour toutes les lignes d'une table, il ne peut pas y avoir de valeurs en double dans une colonne. Alternativement, lorsque plusieurs colonnes sont combinées dans une contrainte unique, la combinaison des valeurs de ces colonnes ne doit pas être dupliquée. En d'autres termes, pour la ou les colonnes uniques spécifiées, chaque ligne doit être distincte.

Contrainte de clé primaire En termes de données autorisées et interdites par une contrainte, une contrainte de clé primaire est identique à une contrainte unique.

Contrainte non nulle Spécifie qu'une colonne ne peut pas stocker de valeur `NULL` et, par conséquent, qu'elle doit avoir une valeur pour chaque ligne.

Contrainte de vérification Permet de spécifier une contrainte arbitraire pour une ou plusieurs tables. La règle qui définit que la valeur d'une colonne doit être comprise entre certaines limites est une contrainte de vérification courante (par exemple que la valeur d'une colonne numérique doit être supérieure à 0). Un autre type courant de contrainte de vérification spécifie les relations entre les valeurs des colonnes (par exemple que la valeur d'une colonne doit être différente de la valeur d'une autre colonne pour la même ligne).

Contrainte de type de données (affinité des colonnes) Le moteur d'exécution impose le type de données des valeurs des colonnes et une erreur se produit en cas de tentative de stockage d'une valeur de type incorrect dans une colonne. Toutefois, les valeurs sont très souvent converties de manière à correspondre au type de données déclaré de la colonne. Pour plus d'informations, voir la section « Utilisation des types de données des bases de données » à la page 781.

Le moteur d'exécution n'impose pas de contrainte sur les valeurs des clés étrangères. En d'autres termes, ces valeurs de clé étrangère ne doivent pas obligatoirement correspondre à une valeur de clé primaire existante.

Outre les types de contraintes prédéfinies, le moteur d'exécution SQL prend en charge l'utilisation de déclencheurs. Un déclencheur est semblable à un gestionnaire d'événement. Il s'agit d'un jeu d'instructions prédéfini qui est exécuté lorsqu'une certaine action se produit. Par exemple, un déclencheur peut être défini pour s'exécuter lorsque des données sont insérées ou supprimées dans une table particulière. Une utilisation possible d'un déclencheur consiste à examiner les modifications apportées aux données et à provoquer une erreur lorsque les conditions spécifiées ne sont pas satisfaites. Ainsi, un déclencheur peut avoir le même objectif qu'une contrainte, et les stratégies qui permettent d'éviter les erreurs et de récupérer à partir d'erreurs de contrainte s'appliquent également aux erreurs générées par les déclencheurs. Toutefois, l'identificateur des erreurs générées par les déclencheurs diffère de celui des erreurs de contrainte.

L'ensemble des contraintes s'appliquant à une table particulière est déterminé lors de la conception d'une application. La conception soignée des contraintes simplifie la conception de l'application quand au fait d'éviter et de récupérer à partir d'erreurs de contrainte. Les erreurs de contrainte sont toutefois difficiles à prévoir et à éviter systématiquement. Les anticipations sont difficiles car les erreurs de contrainte n'apparaissent pas avant l'ajout de données dans l'application. Des erreurs de contraintes surviennent lorsque des données sont ajoutées dans une base de données après sa création. Ces erreurs sont souvent dues aux relations entre les nouvelles données et les données existantes dans la base de données. Les stratégies suivantes permettent d'éviter de nombreuses erreurs de contrainte :

Planification rigoureuse des contraintes et de la structure de la base de données L'objectif des contraintes est d'imposer des règles d'application et de protéger l'intégrité des données de la base de données. Lorsque vous planifiez votre application, prévoyez la structure que doit avoir votre base de données pour prendre en charge cette application. Dans le cadre de ce processus, identifiez les règles devant s'appliquer à vos données, par exemple si certaines valeurs sont obligatoires, si une valeur est définie par défaut, si les valeurs en double sont autorisées, etc. Ces règles vous guident dans la définition des contraintes des bases de données.

Définition explicite des noms des colonnes Il est possible d'écrire une instruction `INSERT` sans spécifier de façon explicite les colonnes dans lesquelles les valeurs doivent être insérées, mais cette technique comporte un risque inutile. En nommant explicitement les colonnes dans lesquelles des valeurs doivent être insérées, vous pouvez autoriser des valeurs générées automatiquement, des colonnes avec des valeurs par défaut et des colonnes autorisant les valeurs `NULL`. Cela vous permet en outre de vous assurer qu'une valeur explicite est insérée dans toutes les colonnes `NOT NULL`.

Utilisation de valeurs par défaut Chaque fois que vous spécifiez une contrainte `NOT NULL` pour une colonne, spécifiez autant que possible une valeur par défaut dans la définition de la colonne. Le code de l'application peut également fournir des valeurs par défaut. Par exemple, votre application peut vérifier si une variable String est `null` et lui affecter une valeur avant de l'utiliser pour définir une valeur de paramètre d'instruction.

Validation des données saisies par l'utilisateur Vérifiez les données saisies par l'utilisateur en amont pour vous assurer qu'elles respectent les contraintes, en particulier dans le cas des contraintes `NOT NULL` et `CHECK`. Bien évidemment, une contrainte `UNIQUE` est plus difficile à vérifier puisqu'elle demanderait l'exécution d'une requête `SELECT` pour déterminer si les données sont uniques.

Utilisation de déclencheurs Vous pouvez écrire un déclencheur qui valide (et éventuellement remplace) les données insérées ou prend d'autres mesures pour corriger les données non valides. Cette validation et cette correction permettent d'éviter les erreurs de contrainte.

Les erreurs de contrainte sont dans tous les cas plus difficiles à prévenir que les autres types d'erreurs. Bien heureusement, plusieurs stratégies permettent de récupérer à partir d'erreurs de contrainte sans affecter la stabilité de l'application ni la rendre inutilisable :

Utilisation d'algorithmes de conflit Lorsque vous définissez une contrainte sur une colonne et que vous créez une instruction `INSERT` ou `UPDATE`, vous avez la possibilité de spécifier un algorithme de conflit. Un algorithme de conflit définit la mesure que la base de données doit prendre en cas de violation d'une contrainte. Le moteur de bases de données peut avoir le choix entre plusieurs actions possibles. Il peut mettre fin à une seule instruction ou à l'ensemble d'une transaction. Il peut ignorer l'erreur. Il peut même supprimer les anciennes données et les remplacer par celles que le code tente de stocker.

Pour plus d'informations, voir la section « ON CONFLICT (algorithmes de conflit) » dans « Prise en charge de SQL dans les bases de données locales » à la page 1154.

Rédaction de commentaires de correction L'ensemble des contraintes susceptibles d'affecter une commande SQL particulière peut être identifié en amont. Vous pouvez par conséquent anticiper les erreurs de contrainte pouvant se produire avec chaque instruction. Ces connaissances vous permettent de développer une logique d'application répondant à une erreur de contrainte. Par exemple, supposons qu'une application comprenne un formulaire de saisie de données pour l'entrée de nouveaux produits. Si la colonne du nom des produits de la base de données est définie avec une contrainte `UNIQUE`, l'insertion d'une nouvelle ligne de produit dans la base de données peut provoquer une erreur de contrainte. Par conséquent, l'application est conçue pour anticiper une telle erreur. Lorsque l'erreur se produit, l'application avertit l'utilisateur, en lui indiquant que le nom du produit spécifié est déjà utilisé et en l'invitant à choisir un autre nom. Une autre réponse possible consiste à autoriser l'utilisateur à consulter les informations relatives au produit portant le même nom.

Utilisation des types de données des bases de données

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'une table est créée dans une base de données, l'instruction de création définit l'affinité, ou le type de données, pour chaque colonne de cette table. Bien que les déclarations d'affinité puissent être omises, il est préférable de déclarer explicitement l'affinité des colonnes dans vos instructions SQL `CREATE TABLE`.

En règle générale, tout objet stocké dans une base de données par une instruction `INSERT` est renvoyé sous forme d'occurrence du même type de données lorsque vous exécutez une instruction `SELECT`. Toutefois, le type de données de la valeur récupérée peut différer selon l'affinité de la colonne de la base de données dans laquelle la valeur est stockée. Lorsqu'une valeur est stockée dans une colonne, si son type de données ne correspond pas à l'affinité de la colonne, la base de données tente de convertir cette valeur en fonction de cette affinité. Par exemple, si une colonne de base de données est déclarée avec une affinité `NUMERIC`, la base de données tente de convertir les données insérées en classe de stockage numérique (`INTEGER` ou `REAL`) avant de stocker les données. Si les données ne peuvent pas être converties, la base de données renvoie une erreur. Selon cette règle, si la chaîne « 12345 » est insérée dans une colonne `NUMERIC`, la base de données la convertit automatiquement en la valeur entière 12345 avant de la stocker dans la base de données. Lorsqu'elle est récupérée par une instruction `SELECT`, la valeur est renvoyée sous forme d'occurrence d'un type de données numérique (tel que Number) plutôt que sous la forme d'une occurrence de String.

Le meilleur moyen d'éviter une conversion non désirable du type de données est de respecter deux règles. D'abord, définissez chaque colonne avec l'affinité correspondant au type de données à stocker. Ensuite, insérez uniquement les valeurs dont le type de données correspond à l'affinité définie. Le respect de ces règles présente deux avantages. Les données ne sont pas converties de façon imprévue lors de leur insertion (ce qui peut éventuellement en modifier le sens). De plus, lorsque vous récupérez les données, elles sont renvoyées avec leur type de données d'origine.

Pour plus d'informations sur les types d'affinités de colonne disponibles et l'utilisation des types de données dans une instruction SQL, voir « Prise en charge des types de données » à la page 1177.

Utilisation des opérations de base de données synchrones et asynchrones

Adobe AIR 1.0 et les versions ultérieures

Les sections précédentes ont décrit les opérations de bases de données courantes, telles que la récupération, l'insertion, la mise à jour et la suppression de données, ainsi que la création d'un fichier de bases de données, de tables et d'autres objets dans une base de données. Les exemples montraient comment effectuer ces opérations de façon asynchrone et synchrone.

Nous vous rappelons qu'en mode d'exécution asynchrone, vous demandez au moteur de base de données d'effectuer une opération. Celui-ci travaille ensuite en arrière-plan pendant que l'application poursuit son exécution. Lorsque l'opération est terminée, le moteur de base de données déclenche un événement pour vous le signaler. Le principal avantage de l'exécution asynchrone est que le moteur d'exécution effectue les opérations de base de données en arrière-plan pendant que l'application principale poursuit son exécution. Cela est d'autant plus précieux lorsque l'exécution de l'opération prend beaucoup de temps.

D'un autre côté, les opérations en mode d'exécution synchrone ne s'exécutent pas en arrière-plan. Vous indiquez au moteur de base de données d'effectuer une opération. Le code s'interrompt à ce stade pendant que le moteur de base de données effectue son travail. Lorsque l'opération est terminée, l'exécution se poursuit avec la ligne suivante de votre code.

Une seule connexion de base de données ne permet pas d'exécuter certaines opérations ou instructions de façon synchrone et d'autres de façon asynchrone. Vous précisez si une occurrence de SQLConnection s'exécute en mode synchrone ou asynchrone lors de l'ouverture de la connexion à la base de données. Si vous appelez `SQLConnection.open()`, la connexion opère en mode d'exécution synchrone, et si vous appelez `SQLConnection.openAsync()`, le mode d'exécution asynchrone est utilisé. Dès qu'une occurrence de SQLConnection est connectée à une base de données par une méthode `open()` ou `openAsync()`, elle fonctionne définitivement en mode synchrone ou asynchrone.

Utilisation des opérations de base de données synchrones

Adobe AIR 1.0 et les versions ultérieures

Le code utilisé pour exécuter et répondre aux opérations en mode d'exécution synchrone et celui utilisé en mode exécution asynchrone ne présentent que peu de différences. Les principales différences entre les deux approches sont de deux types. Le premier est l'exécution d'une opération qui dépend d'une autre opération (telles que les lignes de résultat de `SELECT` ou la clé primaire d'une ligne ajoutée par une instruction `INSERT`). Le second type de différence est la gestion des erreurs.

Ecriture de code pour les opérations synchrones

Adobe AIR 1.0 et les versions ultérieures

La principale différence entre une exécution synchrone et asynchrone est qu'en mode synchrone, la rédaction du code prend la forme d'une suite d'étapes. Par contre, dans le code asynchrone, vous enregistrez des écouteurs d'événement et vous répartissez souvent les opérations entre les méthodes des écouteurs. Lorsqu'une base de données est connectée en mode synchrone, vous pouvez exécuter successivement une série d'opérations de base de données dans un seul bloc de code. L'exemple suivant illustre cette technique :

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, OpenMode.UPDATE);

// start a transaction
conn.begin();

// add the customer record to the database
var insertCustomer:SQLStatement = new SQLStatement();
insertCustomer.sqlConnection = conn;
insertCustomer.text =
    "INSERT INTO customers (firstName, lastName) " +
    "VALUES ('Bob', 'Jones')";
insertCustomer.execute();

var customerId:Number = insertCustomer.getResult().lastInsertRowID;

// add a related phone number record for the customer
var insertPhoneNumber:SQLStatement = new SQLStatement();
insertPhoneNumber.sqlConnection = conn;
insertPhoneNumber.text =
    "INSERT INTO customerPhoneNumbers (customerId, number) " +
    "VALUES (:customerId, '800-555-1234')";
insertPhoneNumber.parameters[":customerId"] = customerId;
insertPhoneNumber.execute();

// commit the transaction
conn.commit();
```

Comme vous pouvez le constater, vous appelez les mêmes méthodes pour effectuer les opérations de base de données, que vous utilisiez le mode synchrone ou asynchrone. Les principales différences entre les deux approches sont l'exécution d'une opération dépendant d'une autre opération et la gestion des erreurs.

Exécution d'une opération dépendant d'une autre opération

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous êtes en mode synchrone, il n'est pas nécessaire d'écrire du code qui écoute un événement pour déterminer la fin d'une opération. Vous pouvez supposer que, lorsque l'opération d'une ligne de code se termine avec succès, l'exécution passe à la ligne de code suivante. Par conséquent, pour effectuer une opération dépendant du succès d'une autre, écrivez simplement le code dépendant immédiatement après l'opération dont il dépend. Par exemple, pour coder une application de sorte qu'elle commence une transaction, exécute une instruction `INSERT`, récupère la clé primaire de la ligne insérée, insère cette clé primaire dans une autre ligne d'une autre table et finisse par valider la transaction, l'ensemble du code peut être écrit sous la forme d'une série d'instructions. L'exemple suivant illustre ces opérations :

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, SQLMode.UPDATE);

// start a transaction
conn.begin();

// add the customer record to the database
var insertCustomer:SQLStatement = new SQLStatement();
insertCustomer.sqlConnection = conn;
insertCustomer.text =
    "INSERT INTO customers (firstName, lastName) " +
    "VALUES ('Bob', 'Jones')";
insertCustomer.execute();

var customerId:Number = insertCustomer.getResult().lastInsertRowID;

// add a related phone number record for the customer
var insertPhoneNumber:SQLStatement = new SQLStatement();
insertPhoneNumber.sqlConnection = conn;
insertPhoneNumber.text =
    "INSERT INTO customerPhoneNumbers (customerId, number) " +
    "VALUES (:customerId, '800-555-1234')";
insertPhoneNumber.parameters[":customerId"] = customerId;
insertPhoneNumber.execute();

// commit the transaction
conn.commit();
```

Gestion des erreurs en mode synchrone

Adobe AIR 1.0 et les versions ultérieures

En mode synchrone, vous n'écoutez pas un événement d'erreur pour déterminer si une opération a échoué. A l'inverse, vous renfermez le code susceptible de déclencher des erreurs dans un jeu de blocs `try..catch..finally`. Vous enveloppez le code rejetant l'erreur dans le bloc `try`. Vous écrivez les actions à effectuer en réponse à chaque type d'erreur dans des blocs `catch` distincts. Vous placez le code qui doit toujours s'exécuter, sans tenir compte de la réussite ou de l'échec (par exemple, la fermeture d'une connexion à la base de données devenue inutile) dans un bloc `finally`. L'exemple suivant démontre l'utilisation des blocs `try..catch..finally` pour la gestion des erreurs. Il développe l'exemple précédent en ajoutant du code de gestion d'erreur :

```
var conn:SQLConnection = new SQLConnection();

// The database file is in the application storage directory
var folder:File = File.applicationStorageDirectory;
var dbFile:File = folder.resolvePath("DBSample.db");

// open the database
conn.open(dbFile, SQLMode.UPDATE);

// start a transaction
conn.begin();

try
{
    // add the customer record to the database
    var insertCustomer:SQLStatement = new SQLStatement();
    insertCustomer.sqlConnection = conn;
    insertCustomer.text =
        "INSERT INTO customers (firstName, lastName)" +
        "VALUES ('Bob', 'Jones')";

    insertCustomer.execute();

    var customerId:Number = insertCustomer.getResult().lastInsertRowID;

    // add a related phone number record for the customer
    var insertPhoneNumber:SQLStatement = new SQLStatement();
    insertPhoneNumber.sqlConnection = conn;
    insertPhoneNumber.text =
        "INSERT INTO customerPhoneNumbers (customerId, number)" +
        "VALUES (:customerId, '800-555-1234')";
    insertPhoneNumber.parameters[":customerId"] = customerId;

    insertPhoneNumber.execute();

    // if we've gotten to this point without errors, commit the transaction
    conn.commit();
}
catch (error:SQLError)
{
    // rollback the transaction
    conn.rollback();
}
```

Présentation du modèle d'exécution asynchrone

Adobe AIR 1.0 et les versions ultérieures

Les développeurs partent souvent du principe qu'en mode asynchrone, il est impossible de lancer l'exécution d'une occurrence de SQLStatement lorsqu'une autre occurrence de SQLStatement est déjà en cours d'exécution sur la même connexion de base de données. En fait, cette hypothèse est fausse. Lorsqu'une occurrence de SQLStatement s'exécute, vous ne pouvez pas modifier la propriété `text` de l'instruction. Toutefois, si vous utilisez une occurrence de SQLStatement distincte pour chaque instruction SQL à exécuter, vous pouvez appeler la méthode `execute()` d'une occurrence de SQLStatement pendant l'exécution d'une autre sans provoquer d'erreur.

Lorsque vous exécutez en interne des opérations de base de données en mode asynchrone, chaque connexion de base de données (chaque occurrence de SQLConnection) possède sa propre file d'attente ou liste d'opérations à exécuter. Le moteur d'exécution effectue chaque opération l'une après l'autre, selon leur ordre d'apparition dans la file d'attente. Lorsque vous créez une occurrence de SQLStatement et appelez sa méthode `execute()`, cette opération d'exécution d'instruction est ajoutée à la file d'attente de la connexion. Si aucune opération n'est en cours d'exécution sur cette occurrence de SQLConnection, l'exécution de l'instruction commence en arrière-plan. Supposons maintenant, que dans le même bloc de code, vous créiez une autre occurrence de SQLStatement et appeliez sa méthode `execute()`. Cette seconde opération d'exécution d'instruction est ajoutée à la file d'attente derrière la première instruction. Dès que l'exécution de la première instruction est terminée, le moteur d'exécution passe à la prochaine opération de la file d'attente. Le traitement des opérations successives de la file d'attente s'effectue en arrière-plan, même lorsque l'événement `result` de la première opération est déclenché dans le code de l'application principale. Le code suivant illustre cette technique :

```
// Using asynchronous execution mode
var stmt1:SQLStatement = new SQLStatement();
stmt1.sqlConnection = conn;

// ... Set statement text and parameters, and register event listeners ...

stmt1.execute();

// At this point stmt1's execute() operation is added to conn's execution queue.

var stmt2:SQLStatement = new SQLStatement();
stmt2.sqlConnection = conn;

// ... Set statement text and parameters, and register event listeners ...

stmt2.execute();

// At this point stmt2's execute() operation is added to conn's execution queue.
// When stmt1 finishes executing, stmt2 will immediately begin executing
// in the background.
```

L'exécution automatique des instructions successives en attente a un effet secondaire important pour la base de données. Lorsqu'une instruction dépend du résultat d'une autre opération, vous ne pouvez pas l'ajouter en file d'attente (en d'autres termes, vous ne pouvez pas appeler sa méthode `execute()`) avant que la première opération ne soit terminée. La raison en est qu'après avoir appelé la méthode `execute()` de la seconde instruction, vous ne pouvez plus modifier les propriétés `text` et `parameters` de l'instruction. Dans ce cas, vous devez attendre l'événement indiquant que la première opération est terminée avant de commencer la suivante. Par exemple, si vous souhaitez exécuter une instruction dans le contexte d'une transaction, l'exécution de cette instruction dépend de l'opération d'ouverture de la transaction. Après l'appel à la méthode `SQLConnection.begin()` pour ouvrir la transaction, vous devez attendre que l'occurrence de SQLConnection déclenche son événement `begin`. A ce stade seulement vous pouvez appeler la méthode `execute()` de l'occurrence de SQLStatement. Dans cet exemple, le moyen le plus simple d'organiser l'application pour s'assurer que les opérations s'exécutent correctement consiste à créer une méthode enregistrée en tant qu'écouteur de l'événement `begin`. Le code qui appelle la méthode `SQLStatement.execute()` est placé dans cette méthode d'écouteur.

Utilisation du chiffrement avec les bases de données SQL

Adobe AIR 1.5 et les versions ultérieures

Toutes les applications Adobe AIR partagent le même moteur de base de données locale. Par conséquent, toute application AIR peut se connecter, lire et écrire dans un fichier d'une base de données non chiffrée. Depuis Adobe AIR 1.5, AIR a la capacité de créer et de se connecter à des fichiers d'une base de données chiffrée. Lorsque vous utilisez une base de données chiffrée, l'application doit fournir la clé de chiffrement appropriée pour se connecter à cette base de données. Si la clé de chiffrement fournie n'est pas correcte (ou s'il n'y a pas de clé), l'application ne peut pas se connecter à la base de données. Elle ne peut donc pas lire les données de la base de données ni y écrire ou modifier des données.

Pour utiliser une base de données chiffrée, vous devez la créer en tant que base de données chiffrée. Avec une base de données chiffrée existante, vous pouvez ouvrir une connexion à la base de données. Vous pouvez également modifier la clé de chiffrement d'une base de données chiffrée. Mises à part la création et la connexion aux bases de données chiffrées, les techniques de travail sont les mêmes que pour une base de données non chiffrée. En particulier, l'exécution des instructions SQL est identique, que la base de données soit chiffrée ou non.

Utilisation d'une base de données chiffrée

Adobe AIR 1.5 et les versions ultérieures

Le chiffrement se révèle très utile dès que vous souhaitez limiter l'accès aux informations stockées dans une base de données. Dans Adobe AIR, la fonction de chiffrement de base de données peut être utilisée dans plusieurs objectifs. Voici quelques exemples pour lesquels l'utilisation d'une base de données chiffrée peut être utile :

- Cache en lecture seule de données d'application privées et téléchargées depuis un serveur
- Stockage d'application local de données privées synchronisées avec un serveur (données envoyées et chargées depuis le serveur)
- Fichiers chiffrés utilisés en tant que format de fichier pour les documents créés et modifiés par l'application Les fichiers peuvent être réservés à un utilisateur, donc privés, ou conçus pour être partagés par tous les utilisateurs de l'application.
- Toute autre utilisation d'un magasin local de données, par exemple les utilisations décrites à la section « Cas d'utilisation des bases de données SQL locales » à la page 741, où les données peuvent être réservées aux personnes disposant d'un accès à l'ordinateur ou aux fichiers de la base de données.

Comprendre la raison pour laquelle vous souhaitez utiliser une base de données chiffrée vous permet de choisir l'architecture de votre application. Le chiffrement risque en particulier d'affecter la manière dont l'application crée, obtient et stocke la clé de chiffrement de la base de données. Pour plus d'informations sur ces considérations, voir « Considérations relatives à l'utilisation du chiffrement avec une base de données » à la page 791.

Parallèlement à l'utilisation d'une base de données chiffrée, le magasin local chiffré permet également de préserver la confidentialité des données sensibles. Grâce au magasin local chiffré, vous stockez une valeur ByteArray unique avec une clé String. Seule l'application AIR qui stocke la valeur peut y accéder, et ceci uniquement sur l'ordinateur sur lequel elle est stockée. Avec le magasin local chiffré, il n'est pas nécessaire de créer votre propre clé de chiffrement. Pour ces raisons, le magasin local chiffré convient davantage au stockage aisé d'une seule valeur ou d'un ensemble de valeurs facilement encodé dans un objet ByteArray. Une base de données chiffrée convient mieux aux grands ensembles de données où l'interrogation et le stockage de données structurées sont souhaitables. Pour plus d'informations sur l'utilisation du magasin local chiffré, voir « Stockage local chiffré » à la page 737.

Création d'une base de données chiffrée

Adobe AIR 1.5 et les versions ultérieures

Pour utiliser une base de données chiffrée, son fichier doit être chiffré lors de sa création. Lorsqu'une base de données a été créée sans chiffrement, elle ne peut plus être chiffrée par la suite. De la même façon, une base de données chiffrée ne peut pas devenir non chiffrée. Si nécessaire, vous pouvez modifier la clé de chiffrement d'une base de données chiffrée. Pour plus d'informations, voir la section « Modification de la clé de chiffrement d'une base de données » à la page 790. Si votre base de données n'est pas chiffrée et que vous souhaitez utiliser le chiffrement de bases de données, vous pouvez créer une nouvelle base de données chiffrée, puis y copier la structure et les données des tables existantes.

La création d'une base de données chiffrée est très similaire à la création d'une base de données non chiffrée, décrite à la section « Création d'une base de données » à la page 746. Vous commencez par créer une occurrence de SQLConnection représentant la connexion à la base de données. Vous créez la base de données en appelant la méthode `open()` ou `openAsync()` de l'objet SQLConnection, en spécifiant un fichier qui n'existe pas encore pour l'emplacement de la base de données. La seule différence lors de la création d'une base de données chiffrée est que vous fournissez une valeur au paramètre `encryptionKey` (cinquième paramètre de la méthode `open()` et sixième paramètre de la méthode `openAsync()`).

Un objet ByteArray contenant exactement 16 octets constitue une valeur valide du paramètre `encryptionKey`.

Les exemples suivants illustrent la création d'une base de données chiffrée. Pour plus de simplicité, la clé de chiffrement est codée en dur dans le code de l'application dans ces exemples. Toutefois, cette technique est fortement déconseillée car elle n'est pas sécurisée.

```
var conn:SQLConnection = new SQLConnection();

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

// Create an encrypted database in asynchronous mode
conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);

// Create an encrypted database in synchronous mode
conn.open(dbFile, SQLMode.CREATE, false, 1024, encryptionKey);
```

Vous trouverez un exemple décrivant un moyen conseillé de générer une clé de chiffrement à la section « Exemple : Génération et utilisation d'une clé de chiffrement » à la page 792.

Connexion à une base de données chiffrée

Adobe AIR 1.5 et les versions ultérieures

Comme la création d'une base de données chiffrée, la procédure qui permet d'ouvrir une connexion à une base de données chiffrée est la même que pour une base de données non chiffrée. Cette procédure est détaillée à la section « Connexion à une base de données » à la page 754. Vous faites appel à la méthode `open()` pour ouvrir une connexion en mode d'exécution synchrone, à la méthode `openAsync()` pour ouvrir une connexion en mode d'exécution asynchrone. La seule différence est que, pour ouvrir une base de données chiffrée, vous spécifiez la valeur correcte du paramètre `encryptionKey` (cinquième paramètre de la méthode `open()` et sixième paramètre de la méthode `openAsync()`).

Si la clé de chiffrement fournie n'est pas correcte, une erreur se produit. Dans le cas de la méthode `open()`, une exception SQLError est renvoyée. Dans le cas de la méthode `openAsync()`, l'objet SQLConnection distribue un événement SQLErrorEvent, dont la propriété `error`contient un objet SQLError. Dans les deux cas, l'objet SQLError généré par l'exception a une valeur de propriété `errorID` 3138. Cet ID d'erreur correspond au message d'erreur « Le fichier ouvert n'est pas un fichier de base de données ».

L'exemple suivant décrit l'ouverture d'une base de données chiffrée en mode d'exécution asynchrone. Pour plus de simplicité, dans cet exemple, la clé de chiffrement est codée en dur dans le code de l'application. Toutefois, cette technique est fortement déconseillée car elle n'est pas sécurisée.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.events.SQLErrorEvent;
import flash.events.SQLEvent;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);
var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

conn.openAsync(dbFile, SQLMode.UPDATE, null, false, 1024, encryptionKey);

function openHandler(event:SQLEvent):void
{
    trace("the database opened successfully");
}

function errorHandler(event:SQLErrorEvent):void
{
    if (event.error.errorID == 3138)
    {
        trace("Incorrect encryption key");
    }
    else
    {
        trace("Error message:", event.error.message);
        trace("Details:", event.error.details);
    }
}
```

L'exemple suivant décrit l'ouverture d'une base de données chiffrée en mode d'exécution synchrone. Pour plus de simplicité, dans cet exemple, la clé de chiffrement est codée en dur dans le code de l'application. Toutefois, cette technique est fortement déconseillée car elle n'est pas sécurisée.

```
import flash.data.SQLConnection;
import flash.data.SQLMode;
import flash.filesystem.File;

var conn:SQLConnection = new SQLConnection();
var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");

var encryptionKey:ByteArray = new ByteArray();
encryptionKey.writeUTFBytes("Some16ByteString"); // This technique is not secure!

try
{
    conn.open(dbFile, SQLMode.UPDATE, false, 1024, encryptionKey);
    trace("the database was created successfully");
}
catch (error:SQLError)
{
    if (error.errorID == 3138)
    {
        trace("Incorrect encryption key");
    }
    else
    {
        trace("Error message:", error.message);
        trace("Details:", error.details);
    }
}
```

Vous trouverez un exemple décrivant un moyen conseillé de générer une clé de chiffrement à la section « Exemple : Génération et utilisation d'une clé de chiffrement » à la page 792.

Modification de la clé de chiffrement d'une base de données

Adobe AIR 1.5 et les versions ultérieures

Vous pouvez modifier ultérieurement la clé de chiffrement d'une base de données chiffrée. Pour modifier la clé de chiffrement d'une base de données, commencez par ouvrir une connexion à la base de données en créant une occurrence de SQLConnection, puis appelez sa méthode `open()` ou `openAsync()`. Lorsque la connexion à la base de données est établie, appelez la méthode `reencrypt()` en transmettant la nouvelle clé de chiffrement en tant qu'argument.

Comme la plupart des opérations de base de données, le comportement de la méthode `reencrypt()` varie selon si la connexion à la base de données utilise le mode d'exécution synchrone ou asynchrone. Si vous utilisez la méthode `open()` pour vous connecter à la base de données, l'opération `reencrypt()` s'exécute de façon synchrone. Lorsque l'opération est terminée, l'exécution se poursuit avec la ligne suivante du code :

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

À l'inverse, si la connexion à la base de données est établie avec la méthode `openAsync()`, l'opération `reencrypt()` est asynchrone. L'appel à la méthode `reencrypt()` commence le processus de rechiffrement. Lorsque l'opération est terminée, l'objet SQLConnection distribue un événement `reencrypt`. Vous utilisez un écouteur d'événement pour connaître le moment où le rechiffrement se termine :

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey

conn.addEventListener(SQLEvent.REENCRYPT, reencryptHandler);

conn.reencrypt(newKey);

function reencryptHandler(event:SQLEvent):void
{
    // save the fact that the key changed
}
```

L'opération `reencrypt()` s'exécute dans sa propre transaction. Si l'opération est interrompue ou échoue (par exemple, si l'application est fermée avant la fin de l'opération), la transaction est annulée. Dans ce cas, la clé de chiffrement d'origine demeure en vigueur pour la base de données.

La méthode `reencrypt()` ne peut pas être utilisée pour supprimer le chiffrage d'une base de données. La transmission d'une valeur `null` ou d'une clé de chiffrement qui ne correspond pas à un objet ByteArray de 16 octets à la méthode `reencrypt()` provoque une erreur.

Considérations relatives à l'utilisation du chiffrement avec une base de données

Adobe AIR 1.5 et les versions ultérieures

La section « Utilisation d'une base de données chiffrée » à la page 787 présente plusieurs cas de figure pour lesquels l'utilisation d'une base de données chiffrée peut être nécessaire. À l'évidence, les besoins de confidentialité diffèrent selon les différents cas d'utilisation des applications (les scénarios présentés, ainsi que d'autres). L'architecture du chiffrement dans votre application joue un rôle important sur le contrôle de la confidentialité des données d'une base de données. Par exemple, si vous utilisez une base de données chiffrée pour assurer la confidentialité des données personnelles, même vis-à-vis des autres utilisateurs de l'ordinateur, la base de données de chaque utilisateur a besoin de sa propre clé de chiffrement. Pour une sécurité optimale, votre application peut générer la clé à partir d'un mot de passe saisi par l'utilisateur. Baser la clé de chiffrement sur un mot de passe permet d'être certain que les données demeurent inaccessibles, même si une autre personne utilise le compte de l'utilisateur sur l'ordinateur. À l'extrême opposé, supposons à présent que vous souhaitiez qu'un fichier de bases de données soit lisible par tout utilisateur de votre application, mais pas par les utilisateurs d'autres applications. Dans ce cas, chaque copie installée de l'application doit pouvoir accéder à une clé de chiffrement partagée.

Vous pouvez concevoir votre application, et en particulier la technique utilisée pour générer la clé de chiffrement, en fonction du niveau de confidentialité désiré pour les données de l'application. Voici une liste de quelques suggestions de conception répondant à divers niveaux de confidentialité des données :

- Pour qu'une base de données soit accessible à tout utilisateur autorisé à accéder à l'application sur n'importe quel ordinateur, utilisez une seule clé disponible pour toutes les occurrences de l'application. Par exemple, lors de sa première exécution, l'application peut télécharger la clé de chiffrement partagée sur un serveur en utilisant un protocole sécurisé de type SSL. Elle peut ensuite enregistrer la clé dans le magasin local chiffré pour l'utiliser ultérieurement. Une autre alternative consiste à chiffrer les données par utilisateur sur l'ordinateur, puis de synchroniser ces données à l'aide d'un magasin de données distant, tel qu'un serveur, pour rendre les données portables.

- Pour qu'un seul utilisateur puisse accéder à la base de données sur n'importe quel ordinateur, générez la clé de chiffrement en utilisant un secret propre à l'utilisateur (par exemple un mot de passe). En particulier, n'utilisez pas de valeur associée à un ordinateur particulier (par exemple une valeur stockée dans le magasin local chiffré) pour générer la clé. Une autre alternative consiste à chiffrer les données par utilisateur sur l'ordinateur, puis de synchroniser ces données à l'aide d'un magasin de données distant, tel qu'un serveur, pour rendre les données portables.
- Pour qu'un seul utilisateur puisse accéder à la base de données sur un seul ordinateur, générez la clé à partir d'un mot de passe et d'une valeur générée arbitrairement, appelée salt. Un exemple de cette technique est disponible à la section « Exemple : Génération et utilisation d'une clé de chiffrement » à la page 792.

Voici d'autres considérations relatives à la sécurité qu'il est important de ne pas oublier lors de la conception d'une application utilisant une base de données chiffrée :

- Un système est autant sécurisé que ce que peut l'être son lien le plus faible. Si vous utilisez un mot de passe saisi par l'utilisateur pour générer une clé de chiffrement, pensez à imposer une longueur minimale et des contraintes de complexité pour les mots de passe. Un mot de passe court qui utilise seulement des caractères de base est facile à deviner.
- Le code source d'une application AIR est stocké sur l'ordinateur de l'utilisateur en texte brut (dans le cas de contenu HTML) ou dans un format binaire facile à décompiler (dans le cas de contenu SWF). Le code source étant accessible, les deux éléments à ne pas oublier sont :
  - Ne jamais coder en dur la clé de chiffrement dans votre code source
  - Toujours partir du principe qu'un attaquant peut aisément découvrir la technique utilisée pour générer une clé de chiffrement (par exemple un générateur de caractères aléatoire ou un algorithme de hachage particulier)
- Le chiffrement de base de données d'AIR utilise le chiffrement AES (Advanced Encryption Standard) avec le mode CBC-MAC (CCM). Pour être sécurisé, ce chiffrement combine la clé saisie par l'utilisateur avec une valeur salt. Un exemple de cette technique est disponible à la section « Exemple : Génération et utilisation d'une clé de chiffrement » à la page 792.
- Lorsque vous choisissez de chiffrer une base de données, tous les fichiers disque utilisés par le moteur de base de données en combinaison avec celle-ci sont chiffrés. Toutefois, le moteur de base de données stocke certaines données temporaires en mémoire cache pour améliorer les performances en lecture et écriture au cours des transactions. Toutes les données qui résident en mémoire ne sont pas chiffrées. Si un attaquant peut accéder à la mémoire utilisée par l'application AIR, par exemple avec un débogueur, les données de la base de données ouverte et non chiffrée sont disponibles.

Exemple : Génération et utilisation d'une clé de chiffrement

Adobe AIR 1.5 et les versions ultérieures

Cet exemple d'application présente une technique de génération de clé de chiffrement. Cette application est conçue pour assurer le plus haut niveau de confidentialité et de sécurité aux données des utilisateurs. Un point essentiel de la sécurisation des données privées consiste à obliger l'utilisateur à saisir un mot de passe à chaque connexion de l'application à la base de données. Par conséquent, comme nous l'avons vu dans cet exemple, une application exigeant ce niveau de confidentialité ne doit jamais stocker directement la clé de chiffrement de la base de données.

L'application comprend deux parties : une classe ActionScript qui génère une clé de chiffrement (la classe EncryptionKeyGenerator) et une interface utilisateur de base qui décrit l'utilisation de cette classe. Pour obtenir le code source complet, voir la section « Exemple de code complet pour la génération et l'utilisation d'une clé de chiffrement » à la page 795.

Utilisation de la classe EncryptionKeyGenerator pour obtenir une clé de chiffrement sécurisée

Adobe AIR 1.5 et les versions ultérieures

Il n'est pas nécessaire de maîtriser les détails du fonctionnement de la classe EncryptionKeyGenerator pour l'utiliser dans votre application. Si vous souhaitez savoir comment la classe génère une clé de chiffrement pour une base de données, voir « Fonctionnement de la classe EncryptionKeyGenerator » à la page 799.

Pour utiliser la classe EncryptionKeyGenerator dans votre application, procédez comme suit :

1. Téléchargez la classe EncryptionKeyGenerator en tant que code source ou SWC compilé. La classe EncryptionKeyGenerator est incluse dans le projet de bibliothèque centrale Open Source ActionScript 3.0 (as3corelib). Vous pouvez télécharger le package as3corelib comprenant le code source et la documentation. Vous pouvez également télécharger des fichiers du code source ou SWC depuis la page du projet.
2. Placez le code source de la classe EncryptionKeyGenerator (ou le fichier SWC as3corelib) dans un emplacement accessible au code source de votre application.
3. Dans le code source de votre application, ajoutez une instruction `import` pour la classe EncryptionKeyGenerator.

   ```
   import com.adobe.air.crypto.EncryptionKeyGenerator;
   ```

4. Avant l'endroit où le code crée la base de données ou ouvre une connexion vers celle-ci, ajoutez le code qui crée une occurrence EncryptionKeyGenerator en appelant le constructeur `EncryptionKeyGenerator()`.

   ```
   var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
   ```

5. Demandez le mot de passe à l'utilisateur :

   ```
   var password:String = passwordInput.text;

   if (!keyGenerator.validateStrongPassword(password))
   {
       // display an error message
       return;
   }
   ```

   L'occurrence de EncryptionKeyGenerator utilise ce mot de passe comme base de la clé de chiffrement (décrite à l'étape suivante). L'occurrence de EncryptionKeyGenerator teste le mot de passe par rapport aux exigences de validation de mot de passe renforcé. Si la validation échoue, une erreur survient. Comme le montre l'exemple de code, vous pouvez vérifier le mot de passe en amont en appelant la méthode `validateStrongPassword()` de l'objet EncryptionKeyGenerator. Cette opération vous permet de déterminer si le mot de passe répond aux exigences minimales d'un mot de passe renforcé et d'éviter une erreur.

6. Générez la clé de chiffrement à partir du mot de passe :

   ```
   var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
   ```

   La méthode `getEncryptionKey()` génère et renvoie la clé de chiffrement (objet ByteArray à 16 octets). Vous pouvez alors utiliser la clé de chiffrement pour créer votre nouvelle base des données chiffrée ou ouvrir une base de données existante.

   La méthode `getEncryptionKey()` a un paramètre obligatoire, le mot de passe obtenu à l'étape 5.

Remarque : *pour assurer le plus haut niveau de sécurité et de confidentialité des données, l'application doit obliger l'utilisateur à saisir un mot de passe à chaque connexion de l'application à la base de données. Ne stockez pas directement le mot de passe de l'utilisateur ni la clé de chiffrement de la base de données. Cela entraînerait des risques de sécurité. Au contraire, comme le montre cet exemple, l'application doit utiliser la même technique pour faire dériver la clé de chiffrement du mot de passe lors de la création de la base des données chiffrée et lors des connexions ultérieures à celle-ci.*

La méthode `getEncryptionKey()` accepte également un second paramètre (facultatif) `overrideSaltELSKey`. L'occurrence de EncryptionKeyGenerator crée une valeur aléatoire (appelée valeur *salt*) qui fait partie de la clé de chiffrement. Pour pouvoir recréer la clé de chiffrement, la valeur salt est stockée dans le magasin local chiffré (ELS) de votre application AIR. Par défaut, la classe EncryptionKeyGenerator utilise une chaîne particulière en tant que clé ELS. Bien que cela soit improbable, il est possible que cette clé soit en conflit avec une autre clé utilisée par votre application. Au lieu d'utiliser la clé par défaut, vous pouvez spécifier votre propre clé ELS. Dans ce cas, spécifiez une clé personnalisée en la transmettant en tant que second paramètre `getEncryptionKey()`, comme illustré ici :

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 Création ou ouverture de la base de données

Avec la clé de chiffrement renvoyée par la méthode `getEncryptionKey()`, votre code peut créer une nouvelle base de données chiffrée ou tenter d'ouvrir la base de données chiffrée existante. Dans les deux cas, vous utilisez la méthode `open()` ou `openAsync()` de la classe SQLConnection, tel que décrit dans les sections « Création d'une base de données chiffrée » à la page 788 et « Connexion à une base de données chiffrée » à la page 788.

Dans cet exemple, l'application est conçue pour ouvrir la base de données en mode d'exécution asynchrone. Le code configure les écouteurs d'événement appropriés et appelle la méthode `openAsync()`, en transmettant la clé de chiffrement en tant qu'argument final :

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

Dans les méthodes des écouteurs, le code supprime les enregistrements des écouteurs d'événement. Il affiche ensuite un message d'état indiquant si la base de données a été créée, ouverte, ou si une erreur s'est produite. La partie la plus importante de ces gestionnaires d'événement se trouve dans la méthode `openError()`. Dans cette méthode, une instruction `if` vérifie l'existence de la base de données (c'est-à-dire que le code tente de se connecter à une base de données existante) et si l'ID d'erreur correspond à la constante `EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID`. Si ces deux conditions sont vraies, il est probable que le mot de passe saisi par l'utilisateur soit incorrect. (Cela peut également signifier que le fichier spécifié n'est pas un fichier de base de données.) Voici le code qui vérifie l'ID d'erreur :

```
if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

Vous trouverez le code complet des exemples d'écouteurs d'événement à la section « Exemple de code complet pour la génération et l'utilisation d'une clé de chiffrement » à la page 795.

Exemple de code complet pour la génération et l'utilisation d'une clé de chiffrement

Adobe AIR 1.5 et les versions ultérieures

Voici le code complet d'un exemple d'application « Génération et utilisation d'une clé de chiffrement » : Le code comprend deux parties.

L'exemple utilise la classe EncryptionKeyGenerator pour créer une clé de chiffrement à partir d'un mot de passe. La classe EncryptionKeyGenerator est incluse dans le projet de bibliothèque centrale Open Source ActionScript 3.0 (as3corelib). Vous pouvez télécharger le package as3corelib comprenant le code source et la documentation. Vous pouvez également télécharger des fichiers du code source ou SWC depuis la page du projet.

Exemple Flex

Le fichier MXML de l'application contient le code source d'une application simple qui crée ou ouvre une connexion à une base de données chiffrée :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
creationComplete="init();">
    <mx:Script>
        <![CDATA[
            import com.adobe.air.crypto.EncryptionKeyGenerator;

            private const dbFileName:String = "encryptedDatabase.db";

            private var dbFile:File;
            private var createNewDB:Boolean = true;
            private var conn:SQLConnection;

            // ------- Event handling -------

            private function init():void
            {
                conn = new SQLConnection();
                dbFile = File.applicationStorageDirectory.resolvePath(dbFileName);
                if (dbFile.exists)
                {
                    createNewDB = false;
                    instructions.text = "Enter your database password to open the encrypted
database.";
                    openButton.label = "Open Database";
                }
            }

            private function openConnection():void
            {
                var password:String = passwordInput.text;

                var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();

                if (password == null || password.length <= 0)
                {
                    statusMsg.text = "Please specify a password.";
                    return;
                }
```

```
                if (!keyGenerator.validateStrongPassword(password))
                {
                    statusMsg.text = "The password must be 8-32 characters long. It must
contain at least one lowercase letter, at least one uppercase letter, and at least one number
or symbol.";
                    return;
                }

                passwordInput.text = "";
                passwordInput.enabled = false;
                openButton.enabled = false;

                var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, openError);

                conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
            }

            private function openHandler(event:SQLEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                statusMsg.setStyle("color", 0x009900);
                if (createNewDB)
                {
                    statusMsg.text = "The encrypted database was created successfully.";
                }
                else
                {
                    statusMsg.text = "The encrypted database was opened successfully.";
                }
            }

            private function openError(event:SQLErrorEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                if (!createNewDB && event.error.errorID ==
```

```
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
                {
                    statusMsg.text = "Incorrect password!";
                }
                else
                {
                    statusMsg.text = "Error creating or opening database.";
                }
            }
        ]]>
    </mx:Script>
    <mx:Text id="instructions" text="Enter a password to create an encrypted database. The next
time you open the application, you will need to re-enter the password to open the database
again." width="75%" height="65"/>
    <mx:HBox>
        <mx:TextInput id="passwordInput" displayAsPassword="true"/>
        <mx:Button id="openButton" label="Create Database" click="openConnection();"/>
    </mx:HBox>
    <mx:Text id="statusMsg" color="#990000" width="75%"/>
</mx:WindowedApplication>
```

Exemple Flash Professional

Le fichier FLA de l'application contient le code source d'une application simple qui crée ou ouvre une connexion à une base de données chiffrée. Le fichier FLA comprend quatre composants placés sur la scène :

Nom d'occurrence	Type de composant	Description
`instructions`	Label	Contient les instructions données à l'utilisateur
`passwordInput`	TextInput	Champ de saisie dans lequel l'utilisateur tape le mot de passe
`openButton`	Button	Bouton sur lequel l'utilisateur clique après avoir saisi le mot de passe
`statusMsg`	Label	Affiche des messages d'état (réussite ou échec)

Le code de l'application est défini sur une image-clé de l'image 1 du scénario principal. Voici le code de l'application :

```
import com.adobe.air.crypto.EncryptionKeyGenerator;

const dbFileName:String = "encryptedDatabase.db";

var dbFile:File;
var createNewDB:Boolean = true;
var conn:SQLConnection;

init();

// ------- Event handling -------

function init():void
{
    passwordInput.displayAsPassword = true;
    openButton.addEventListener(MouseEvent.CLICK, openConnection);
    statusMsg.setStyle("textFormat", new TextFormat(null, null, 0x990000));

    conn = new SQLConnection();
    dbFile = File.applicationStorageDirectory.resolvePath(dbFileName);

    if (dbFile.exists)
    {
        createNewDB = false;
        instructions.text = "Enter your database password to open the encrypted database.";
        openButton.label = "Open Database";
    }
    else
    {
        instructions.text = "Enter a password to create an encrypted database. The next time
you open the application, you will need to re-enter the password to open the database again.";
        openButton.label = "Create Database";
    }
}

function openConnection(event:MouseEvent):void
{
    var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();

    var password:String = passwordInput.text;

    if (password == null || password.length <= 0)
    {
        statusMsg.text = "Please specify a password.";
        return;
    }

    if (!keyGenerator.validateStrongPassword(password))
    {
        statusMsg.text = "The password must be 8-32 characters long. It must contain at least
one lowercase letter, at least one uppercase letter, and at least one number or symbol.";
        return;
    }

    passwordInput.text = "";
    passwordInput.enabled = false;
    openButton.enabled = false;
```

```
    var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);

    conn.addEventListener(SQLEvent.OPEN, openHandler);
    conn.addEventListener(SQLErrorEvent.ERROR, openError);

    conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
}

function openHandler(event:SQLEvent):void
{
    conn.removeEventListener(SQLEvent.OPEN, openHandler);
    conn.removeEventListener(SQLErrorEvent.ERROR, openError);

    statusMsg.setStyle("textFormat", new TextFormat(null, null, 0x009900));
    if (createNewDB)
    {
        statusMsg.text = "The encrypted database was created successfully.";
    }
    else
    {
        statusMsg.text = "The encrypted database was opened successfully.";
    }
}

function openError(event:SQLErrorEvent):void
{
    conn.removeEventListener(SQLEvent.OPEN, openHandler);
    conn.removeEventListener(SQLErrorEvent.ERROR, openError);

    if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
    {
        statusMsg.text = "Incorrect password!";
    }
    else
    {
        statusMsg.text = "Error creating or opening database.";
    }
}
```

Fonctionnement de la classe EncryptionKeyGenerator

Adobe AIR 1.5 et les versions ultérieures

Il n'est pas nécessaire de comprendre le fonctionnement intrinsèque de la classe EncryptionKeyGenerator pour l'utiliser afin de créer une clé de chiffrement sécurisée pour la base de données de votre application. Le processus d'utilisation de la classe est décrit à la section « Utilisation de la classe EncryptionKeyGenerator pour obtenir une clé de chiffrement sécurisée » à la page 793. Il peut cependant se révéler précieux pour comprendre les techniques utilisées par la classe. Par exemple, vous pourriez vouloir adapter la classe ou intégrer certaines de ses techniques lorsqu'un niveau différent de confidentialité des données est nécessaire.

La classe EncryptionKeyGenerator est incluse dans le projet de bibliothèque centrale Open Source ActionScript 3.0 (as3corelib). Vous pouvez télécharger le package as3corelib comprenant le code source et la documentation. Vous pouvez également afficher le code source dans le site du projet ou le télécharger pour suivre les explications.

Lorsque le code crée une occurrence de EncryptionKeyGenerator et appelle sa méthode `getEncryptionKey()`, plusieurs mesures permettent de s'assurer que seul l'utilisateur autorisé peut accéder aux données. Le processus correspond à la génération d'une clé de chiffrement à partir d'un mot de passe saisi par l'utilisateur avant la création de la base de données ou à la recréation de la clé de chiffrement pour ouvrir la base de données.

Obtention et validation d'un mot de passe renforcé

Adobe AIR 1.5 et les versions ultérieures

Lorsque le code appelle la méthode `getEncryptionKey()`, il transmet un mot de passe sous forme de paramètre. Ce mot de passe est utilisé comme base de la clé de chiffrement. En utilisant des informations que seul l'utilisateur connaît, cette technique permet de s'assurer que seul l'utilisateur qui connaît le mot de passe peut accéder au contenu de la base de données. Même s'il accède au compte de l'utilisateur sur l'ordinateur, l'attaquant ne peut pas accéder à la base de données sans connaître le mot de passe. Pour une sécurité maximale, l'application ne stocke jamais le mot de passe.

Le code d'une application crée une occurrence d'EncryptionKeyGenerator et appelle la méthode `getEncryptionKey()` correspondante en transmettant un mot de passe saisi par l'utilisateur sous forme d'argument (soit la variable `password` dans cet exemple) :

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

La première étape effectuée par la classe EncryptionKeyGenerator lors de l'appel à la méthode `getEncryptionKey()` consiste à vérifier que le mot de passe saisi par l'utilisateur respecte les exigences définies en matière de sécurité. La classe EncryptionKeyGenerator requiert que le mot de passe contienne entre 8 et 32 caractères. Le mot de passe doit combiner des lettres majuscules et minuscules et comprendre au moins un chiffre ou un symbole.

L'expression régulière qui vérifie ce modèle est définie en tant que constante nommée `STRONG_PASSWORD_PATTERN` :

```
private static const STRONG_PASSWORD_PATTERN:RegExp =
/(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

Le code qui vérifie le mot de passe est dans la méthode `validateStrongPassword()` de la classe EncryptionKeyGenerator. Le code se présente comme suit :

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

La méthode `getEncryptionKey()` appelle en interne la méthode `validateStrongPassword()` de la classe EncryptionKeyGenerator et renvoie une exception si le mot de passe n'est pas valide. La méthode `validateStrongPassword()` étant publique, le code de l'application peut vérifier le mot de passe sans appeler la méthode `getEncryptionKey()` pour éviter la production d'erreur.

Extension du mot de passe à 256 bits

Adobe AIR 1.5 et les versions ultérieures

Dans la suite du processus, le mot de passe doit avoir une longueur de 256 bits. Plutôt que de demander à chaque utilisateur de saisir un mot de passe faisant exactement 256 bits (32 caractères), le code crée un mot de passe plus long en répétant ses caractères.

Pour exécuter la tâche de création d'un mot de passe long, la méthode `getEncryptionKey()` appelle la méthode `concatenatePassword()`.

```
var concatenatedPassword:String = concatenatePassword(password);
```

Voici le code de la méthode `concatenatePassword()` :

```
private function concatenatePassword(pwd:String):String
{
    var len:int = pwd.length;
    var targetLength:int = 32;

    if (len == targetLength)
    {
        return pwd;
    }

    var repetitions:int = Math.floor(targetLength / len);
    var excess:int = targetLength % len;

    var result:String = "";

    for (var i:uint = 0; i < repetitions; i++)
    {
        result += pwd;
    }

    result += pwd.substr(0, excess);

    return result;
}
```

Si le mot de passe n'atteint pas 256 bits, le code le concatène pour obtenir une longueur de 256 bits. Si la longueur ne fonctionne pas exactement, la dernière répétition est raccourcie jusqu'à correspondre à 256 bits.

Génération ou récupération d'une valeur salt de 256 bits

Adobe AIR 1.5 et les versions ultérieures

L'étape suivante consiste à obtenir une valeur salt de 256 bits qui sera ensuite combinée au mot de passe dans une étape ultérieure. Une valeur *salt* est une valeur aléatoire ajoutée ou combinée à la valeur saisie par l'utilisateur pour composer le mot de passe. L'utilisation d'une valeur salt avec un mot de passe permet de s'assurer que, même si l'utilisateur choisit un mot du dictionnaire ou un terme courant comme mot de passe, la combinaison mot de passe-plus-salt utilisée par le système est une valeur aléatoire. Cet aspect aléatoire assure une protection contre les attaques de type dictionnaire, dans lesquelles l'attaquant utilise une liste de mots pour tenter de deviner le mot de passe. De plus, la génération de cette valeur salt et son stockage dans le magasin local chiffré permet de l'associer au compte de l'utilisateur sur l'ordinateur dans lequel le fichier de base de données est situé.

Si l'application appelle la méthode `getEncryptionKey()` pour la première fois, le code crée une valeur salt aléatoire de 256 bits. Sinon, le code récupère la valeur salt dans le magasin local chiffré.

La valeur salt est stockée dans une variable nommée `salt`. Le code détermine si une valeur salt a déjà été créée en tentant de la récupérer dans le magasin local chiffré :

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

Si le code crée une nouvelle valeur salt, la méthode `makeSalt()` génère une valeur aléatoire de 256 bits. Comme la valeur est ensuite stockée dans le magasin local chiffré, elle est générée sous forme d'objet ByteArray. La méthode `makeSalt()` utilise la méthode `Math.random()` pour générer la valeur de façon aléatoire. La méthode `Math.random()` ne peut pas générer 256 bits en une seule opération. À la place, le code utilise une boucle pour appeler `Math.random()` à huit reprises. Chaque fois, une valeur uint aléatoire comprise entre 0 et 4294967295 (valeur uint maximale) est générée. La valeur uint est utilisée pour plus de commodité car elle comprend exactement 32 bits. L'écriture de huit valeurs uint dans l'objet ByteArray permet de générer une valeur de 256 bits. Voici le code de la méthode `makeSalt()` :

```
private function makeSalt():ByteArray
{
    var result:ByteArray = new ByteArray;

    for (var i:uint = 0; i < 8; i++)
    {
        result.writeUnsignedInt(Math.round(Math.random() * uint.MAX_VALUE));
    }

    return result;
}
```

Lorsque le code enregistre la valeur salt dans le magasin local chiffré ou tente de l'y récupérer, il a besoin d'une clé String dans laquelle la valeur salt est enregistrée. Sans cette clé, la valeur salt ne peut pas être récupérée. Dans ce cas, la clé de chiffrement ne peut pas être recréée à chaque nouvelle ouverture de la base de données. Par défaut, la classe EncryptionKeyGenerator utilise une clé prédéfinie du magasin local chiffré qui est définie dans la constante `SALT_ELS_KEY`. Au lieu d'utiliser la clé par défaut, le code de l'application peut également spécifier la clé de magasin local chiffré à utiliser dans l'appel à la méthode `getEncryptionKey()`. La clé de magasin local chiffré de la valeur salt par défaut ou spécifiée par l'application est stockée dans une variable nommée `saltKey`. Cette variable est utilisée dans les appels de `EncryptedLocalStore.setItem()` et `EncryptedLocalStore.getItem()`, comme indiqué précédemment.

Combinaison du mot de passe 256 bits et de la valeur salt via l'opérateur XOR

Adobe AIR 1.5 et les versions ultérieures

Le code dispose à présent d'un mot de passe de 256 bits et d'une valeur salt de 256 bits. Il utilise ensuite une opération XOR au niveau du bit pour combiner la valeur salt avec le mot de passe concaténé dans une valeur unique. Cette technique crée un mot de passe de 256 bits à partir de tous les caractères possibles de la plage complète. Ce principe reste vrai même si la saisie du véritable mot de passe est plus probablement constituée des principaux caractères alphanumériques. Cet aspect encore plus aléatoire a l'avantage de créer le plus vaste ensemble possible de mots de passe sans que l'utilisateur ne doive lui-même saisir un mot de passe long et complexe.

Le résultat de l'opération XOR est stocké dans la variable `unhashedKey`. La véritable opération XOR au niveau du bit sur les deux valeurs se produit dans la méthode `xorBytes()` :

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

L'opérateur XOR au niveau du bit (^) prend deux valeurs uint et n'en renvoie qu'une. (Une valeur uint contient 32 bits.) Les valeurs d'entrée transmises en tant qu'arguments à la méthode `xorBytes()` sont une valeur String (le mot de passe) et une valeur ByteArray (la valeur salt). Par conséquent, le code utilise une boucle pour extraire 32 bits de chaque entrée à combiner à l'aide de l'opérateur XOR.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

Dans la boucle, les premiers 32 bits (4 octets) sont extraits du paramètre `passwordString`. Ces bits sont extraits et convertis en valeur uint (`o1`) dans un processus à deux parties. D'abord, la méthode `charCodeAt()` récupère la valeur numérique de chaque caractère. Cette valeur est ensuite décalée jusqu'à la position appropriée dans la valeur uint par l'opérateur de décalage gauche au niveau du bit (`<<`), puis la valeur décalée est ajoutée à `o1`. Par exemple, le premier caractère (`i`) devient les premiers 8 bits grâce à l'opérateur de décalage gauche au niveau du bit (`<<`) qui décale les bits de 24 bits à gauche et affecte cette valeur à `o1`. Le second caractère `(i + 1)` prend la place des seconds 8 bits en décalant sa valeur vers la gauche de 16 bits et en ajoutant le résultat à `o1`. Les valeurs des troisième et quatrième caractères sont ajoutées de la même manière.

```
        // ...

        // Extract 4 bytes from the password string and convert to a uint
        var o1:uint = passwordString.charCodeAt(i) << 24;
        o1 += passwordString.charCodeAt(i + 1) << 16;
        o1 += passwordString.charCodeAt(i + 2) << 8;
        o1 += passwordString.charCodeAt(i + 3);

        // ...
```

La variable `o1` contient à présent 32 bits provenant du paramètre `passwordString`. Ensuite, 32 bits sont extraits du paramètre `salt` via un appel à sa méthode `readUnsignedInt()`. Les 32 bits sont stockés dans la variable uint `o2`.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

Enfin, les deux valeurs de 32 bits (uint) sont combinées par l'opérateur XOR et le résultat est écrit dans un objet ByteArray nommé `result`.

```
        // ...

        var xor:uint = o1 ^ o2;
        result.writeUnsignedInt(xor);
        // ...
```

Lorsque la boucle est terminée, l'objet ByteArray contenant le résultat XOR est renvoyé.

```
        // ...
    }

    return result;
}
```

Hachage de la clé

Adobe AIR 1.5 et les versions ultérieures

Une fois le mot de passe concaténé et la valeur salt combinée, l'étape suivante consiste à sécuriser encore davantage cette valeur en la hachant avec un algorithme SHA-256. Le hachage de la valeur rend encore plus difficile sa rétro-conception par un attaquant.

À ce stade, le code possède un objet ByteArray nommé `unhashedKey` contenant le mot de passe concaténé combiné à la valeur salt. Le projet de bibliothèque ActionScript 3.0 (as3corelib) comprend une classe SHA256 dans le package com.adobe.crypto. La méthode `SHA256.hashBytes()` qui exécute un hachage SHA-256 sur l'objet ByteArray et renvoie une valeur String contenant le résultat du hachage 256 bits sous forme de nombre hexadécimal. La classe EncryptionKeyGenerator utilise la classe SHA256 pour hacher la clé :

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

Extraction de la clé de chiffrement du hachage

Adobe AIR 1.5 et les versions ultérieures

La clé de chiffrement doit être un objet ByteArray faisant exactement 16 octets (128 bits). Le résultat de l'algorithme de hachage SHA-256 fait toujours 256 bits. Par conséquent, l'étape finale consiste à sélectionner 128 bits dans le résultat haché et à les utiliser comme véritable clé de chiffrement.

Dans la classe EncryptionKeyGenerator, le code réduit la clé à 128 bits en appelant la méthode `generateEncryptionKey()`. Il renvoie ensuite le résultat de cette méthode en tant que résultat de la méthode `getEncryptionKey()` :

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

Il n'est pas nécessaire d'utiliser les premiers 128 bits comme clé de chiffrement. Vous pouvez sélectionner une plage de bits à partir d'un point arbitraire, sélectionner tous les autres bits ou sélectionner les bits d'une autre manière. L'important est que le code sélectionne 128 bits distincts, et que ces mêmes 128 bits soient utilisés chaque fois.

Dans ce cas, la méthode `generateEncryptionKey()` utilise la plage de bits commençant au 18ème octet en tant que clé de chiffrement. Comme nous l'avons déjà mentionné, la classe SHA256 renvoie une valeur String contenant un hachage 256 bits sous forme de nombre hexadécimal. Un unique bloc de 128 bits comprend trop d'octets pour qu'ils soient ajoutés en une seule fois à un objet ByteArray. Par conséquent, le code utilise une boucle `for` pour extraire les caractères de la valeur String hexadécimale, les convertit en valeurs numériques réelles et les ajoute à l'objet ByteArray. La valeur String SHA-256 résultante comprend 64 caractères. Une plage de 128 bits correspond à 32 caractères dans la valeur String et chaque caractère représente 4 bits. Le plus petit incrément de données que vous pouvez ajouter à un objet ByteArray correspond à un seul octet (8 bits), ce qui équivaut à deux caractères dans la valeur String `hash`. De ce fait, la boucle compte de 0 à 31 (32 caractères) par incréments de 2 caractères.

Dans la boucle, le code commence par déterminer la position de départ de la paire de caractères en cours. Comme la plage désirée commence au niveau du caractère situé à l'index 17 (18ème octet), la variable `position` est affectée à la valeur de l'itérateur actuel (`i`) plus 17. Le code utilise la méthode `substr()` de l'objet String pour extraire les deux caractères situés au niveau de la position en cours. Ces caractères sont stockés dans la variable `hex`. Ensuite, le code utilise la méthode `parseInt()` pour convertir la valeur String `hex` en une valeur décimale entière. Il stocke cette valeur dans la variable int `byte`. Enfin, le code ajoute la valeur de `byte` à l'objet ByteArray `result` à l'aide de sa méthode `writeByte()`. Lorsque la boucle se termine, l'objet ByteArray `result` contient 16 octets et peut être utilisé comme clé de chiffrement pour la base de données.

```
private function generateEncryptionKey(hash:String):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 2)
    {
        var position:uint = i + 17;
        var hex:String = hash.substr(position, 2);
        var byte:int = parseInt(hex, 16);
        result.writeByte(byte);
    }

    return result;
}
```

Stratégies d'utilisation des bases de données SQL

Adobe AIR 1.0 et les versions ultérieures

Une application peut accéder à une base de données SQL locale et l'exploiter de différentes manières. La conception de l'application peut varier en termes d'organisation du code, d'ordre et de synchronisation d'exécution des opérations, etc. Les techniques choisies peuvent affecter la simplicité du développement de votre application, par exemple, la facilité avec laquelle l'application pourra être modifiée dans les futures mises à jour. Elles peuvent également avoir un impact sur le bon fonctionnement de l'application du point de vue des utilisateurs.

Distribution d'une base de données pré-renseignée

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous utilisez une base de données SQL locale AIR dans votre application, cette dernière attend une base de données présentant une certaine structure de tables, de colonnes, etc. Certaines applications s'attendent également à ce que certaines données soient pré-renseignées dans le fichier de la base de données. Créer la base de données au sein du code de l'application est un moyen de s'assurer que la base de données présente la structure appropriée. Lors du chargement de l'application, celle-ci vérifie la présence de son fichier de base de données dans un emplacement particulier. Si le fichier n'existe pas, l'application exécute un ensemble de commandes pour le créer, crée la structure de la base de données et renseigne les tables avec les données initiales.

Le code qui crée la base de données et ses tables est souvent complexe. Bien souvent, il n'est utilisé qu'une fois au début de la durée de vie d'une application installée, mais augmente tout de même la taille et la complexité de celle-ci. Au lieu de créer la base de données, sa structure et ses données par programmation, vous pouvez distribuer une base de données pré renseignée avec votre application. Pour distribuer une base de données prédéfinie, incluez le fichier de base de données dans le package AIR de l'application.

Comme tous les fichiers inclus dans ce package AIR, un fichier de base de données regroupé est installé dans le répertoire de l'application (celui représenté par la propriété `File.applicationDirectory`). Toutefois, les fichiers de ce répertoire sont en lecture seule. Servez-vous du fichier du package AIR en tant que base de données « modèle ». La première fois que l'utilisateur exécute l'application, copiez le fichier de base de données d'origine dans le « Pointage vers le répertoire de stockage d'une application » à la page 697 de l'utilisateur (ou dans un autre emplacement) et utilisez cette base de données dans l'application.

Recommandations concernant l'utilisation des bases de données SQL locales

Adobe AIR 1.0 et les versions ultérieures

Voici une liste de techniques conseillées qui permettront d'améliorer les performances, la sécurité et la simplicité de maintenance des applications lors de l'utilisation de bases de données SQL locales.

Création préalable des connexions à la base de données

Adobe AIR 1.0 et les versions ultérieures

Même si votre application n'exécute aucune instruction lors de son premier chargement, instanciez un objet SQLConnection et appelez sa méthode `open()` ou `openAsync()` en amont (par exemple après le démarrage initial de l'application) pour éviter les délais lors de l'exécution des instructions. Voir la section « Connexion à une base de données » à la page 754.

Réutilisation des connexions à la base de données

Adobe AIR 1.0 et les versions ultérieures

Si vous accédez à une certaine base de données pendant l'exécution de votre application, gardez une référence à l'occurrence de SQLConnection et réutilisez-la dans toute l'application au lieu de fermer et de rouvrir la connexion. Voir la section « Connexion à une base de données » à la page 754.

Choix préférentiel du mode asynchrone

Adobe AIR 1.0 et les versions ultérieures

Lors de l'écriture du code d'accès aux données, il peut être tentant d'exécuter les opérations de façon synchrone plutôt que de façon asynchrone car cela demande souvent un code plus court et moins complexe. Toutefois, comme nous l'avons vu à la section « Utilisation des opérations de base de données synchrones et asynchrones » à la page 782, les opérations synchrones peuvent affecter les performances de façon évidente pour les utilisateurs et nuire à leur exploitation de l'application. Le temps nécessaire à l'exécution d'une seule opération varie d'une opération à l'autre et notamment en fonction de la quantité de données impliquées. Par exemple, une instruction SQL `INSERT` qui n'ajoute qu'une seule ligne à la base de données prend moins de temps qu'une instruction `SELECT` qui récupère des milliers de lignes de données. Toutefois, lorsque vous utilisez le mode synchrone pour exécuter plusieurs opérations, les opérations sont généralement enchaînées. Même si le temps nécessaire à chaque opération est très court, l'application se bloque jusqu'à ce que toutes les opérations synchrones soient terminées. En résultat, le temps cumulé des différentes opérations peut être suffisant pour bloquer votre application.

Utilisez les opérations asynchrones comme approche habituelle, en particulier dans le cas d'opérations impliquant un grand nombre de lignes. Une technique permet de diviser le traitement des vastes jeux de résultats de l'instruction `SELECT`. Cette technique est décrite à la section « Récupération partielle des résultats d'une instruction SELECT » à la page 770. Cette technique ne peut cependant être utilisée qu'en mode d'exécution asynchrone. Utilisez uniquement les opérations synchrones lorsque vous ne pouvez pas obtenir certaines fonctionnalités avec la programmation

asynchrone, lorsque vous avez pris en compte les baisses de performances que les utilisateurs de votre application rencontreront éventuellement et après avoir testé votre application de manière à connaître l'impact sur les performances. L'utilisation du mode d'exécution asynchrone peut impliquer un codage plus complexe. Toutefois, n'oubliez pas que le code ne doit être écrit qu'une seule fois alors que les utilisateurs emploient l'application de façon répétée, qu'elle soit rapide ou lente.

Dans la plupart des cas, l'utilisation d'une occurrence de SQLStatement distincte pour chaque instruction SQL à exécuter permet de mettre plusieurs opérations SQL en file d'attente à la fois, ce qui, en termes d'écriture du code, rend le code asynchrone similaire au code synchrone. Pour plus d'informations, voir la section « Présentation du modèle d'exécution asynchrone » à la page 785.

Utilisation d'instructions SQL distinctes sans modification de la propriété text de l'occurrence de SQLStatement

Adobe AIR 1.0 et les versions ultérieures

Pour chaque instruction SQL exécutée plusieurs fois dans une application, créez une occurrence de SQLStatement distincte. Servez-vous de cette occurrence de SQLStatement chaque fois que cette commande SQL s'exécute. Supposons par exemple que vous développiez une application comprenant quatre opérations SQL différentes exécutées à plusieurs reprises. Dans ce cas, créez quatre occurrences de SQLStatement distinctes et appelez la méthode `execute()` de chaque instruction pour l'exécuter. Evitez d'utiliser une seule occurrence de SQLStatement pour toutes les instructions SQL, en redéfinissant chaque fois sa propriété `text` avant d'exécuter l'instruction.

Utilisation de paramètres d'instruction

Adobe AIR 1.0 et les versions ultérieures

Utilisez des paramètres pour SQLStatement. Ne concaténez jamais la saisie de l'utilisateur dans le texte de l'instruction. L'utilisation de paramètres sécurise votre application car elle empêche les attaques par injection de code SQL. Il est alors possible d'utiliser des objets dans les requêtes (au lieu d'utiliser uniquement des valeurs littérales SQL). Cela renforce également l'efficacité de l'exécution des instructions car ces dernières peuvent être réutilisées sans qu'il soit nécessaire de les recompiler chaque fois qu'elles sont exécutées. Voir la section « Utilisation de paramètres dans des instructions » à la page 758.

Chapitre 41 : Utilisation de tableaux d'octets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ByteArray vous permet de lire à partir d'un flux de données binaire (qui, pour l'essentiel, correspond à un tableau d'octets) ou d'écrire dans un tel flux. Elle offre un moyen d'accéder aux données au niveau le plus élémentaire. Etant donné que les données informatiques se composent d'octets ou de groupes de 8 bits, la capacité de lire des données en octets implique qu'il est possible d'accéder à des données pour lesquelles aucune classe et aucune méthode d'accès n'existent. La classe ByteArray vous permet d'analyser tous les flux de données (d'une image bitmap à un flux de données transitant par le réseau) au niveau de l'octet.

La méthode `writeObject()` vous permet d'écrire un objet au format AMF (Action Message Format) sérialisé dans une classe ByteArray, tandis que la méthode `readObject()` vous donne la possibilité de lire un objet sérialisé à partir d'une classe ByteArray vers une variable du type de données d'origine. Vous avez la possibilité de sérialiser n'importe quel objet, à l'exception des objets d'affichage, lesquels peuvent être placés dans la liste d'affichage. Vous pouvez également réaffecter des objets sérialisés à des occurrences d'une classe personnalisée si celle-ci est disponible pour le moteur d'exécution. Une fois qu'un objet a été converti au format AMF, vous pouvez le transférer rapidement par le biais d'une connexion réseau ou l'enregistrer dans un fichier.

L'application Adobe® AIR® décrite dans cet exemple lit un fichier .zip pour illustrer le traitement d'un flux d'octets. Elle extrait une liste des fichiers contenus dans le fichier .zip et les écrit dans le poste de travail.

Voir aussi

flash.utils.ByteArray

flash.utils.IExternalizable

Spécification AMF (Action Message Format)

Lecture et écriture d'un objet ByteArray

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe ByteArray fait partie du package flash.utils. Afin de créer un objet ByteArray dans ActionScript 3.0, importez la classe ByteArray et appelez le constructeur, comme décrit dans l'exemple suivant :

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

Méthodes ByteArray

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Tout flux de données significatif est organisé selon un format spécifique que vous pouvez analyser afin d'y trouver les informations souhaitées. Par exemple, un enregistrement dans un fichier d'employé simple, comprendrait très certainement un numéro d'ID, un nom, une adresse, un numéro de téléphone, etc. Un fichier audio MP3 contient une balise ID3 permettant d'identifier le titre, l'auteur, l'album, la date d'édition et le genre du fichier en cours de téléchargement. Le format vous permet de connaître l'ordre dans lequel vous devez vous attendre à recevoir les données du flux de données. Il vous permet de lire le flux d'octets de manière intelligente.

La classe ByteArray comprend plusieurs méthodes destinées à faciliter la lecture depuis un flux de données et l'écriture vers lui. Parmi les méthodes disponibles, on compte celles-ci : `readBytes()` et `writeBytes()`, `readInt()` et `writeInt()`, `readFloat()` et `writeFloat()`, `readObject()` et `writeObject()`, et `readUTFBytes()` et `writeUTFBytes()`. Grâce à ces méthodes, vous pouvez lire des données provenant du flux de données dans des variables de types de données spécifiques et écrire directement depuis ces types de données vers un flux de données binaire.

Par exemple, le code suivant lit un simple tableau de chaînes et de nombres à virgule flottante, puis écrit chaque élément vers un objet ByteArray. La structure du tableau permet au code d'appeler les méthodes ByteArray appropriées (`writeUTFBytes()` et `writeFloat()`) afin d'écrire les données. Le modèle de données répétitif permet de lire le tableau au moyen d'une boucle.

```
 // The following example reads a simple Array (groceries), made up of strings
// and floating-point numbers, and writes it to a ByteArray.

import flash.utils.ByteArray;

// define the grocery list Array
var groceries:Array = ["milk", 4.50, "soup", 1.79, "eggs", 3.19, "bread" , 2.35]
// define the ByteArray
var bytes:ByteArray = new ByteArray();
// for each item in the array
for (var i:int = 0; i < groceries.length; i++) {
        bytes.writeUTFBytes(groceries[i++]); //write the string and position to the next item
        bytes.writeFloat(groceries[i]);// write the float
        trace("bytes.position is: " + bytes.position);//display the position in ByteArray
}
trace("bytes length is: " +  bytes.length);// display the length
```

Propriété position

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété position stocke la position actuelle du pointeur qui indexe la classe ByteArray au cours de la lecture ou de l'écriture. La valeur initiale de la propriété position est égale à 0 (zéro) comme l'illustre l'exemple de code suivant :

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

Lors de la lecture à partir d'une classe ByteArray ou de l'écriture vers elle, la méthode utilisée met à jour la propriété position de sorte qu'elle pointe vers l'emplacement suivant immédiatement le dernier octet lu ou écrit. Par exemple, le code suivant écrit une chaîne dans une classe ByteArray. Ensuite, la propriété position pointe vers l'octet qui suit immédiatement la chaîne dans ByteArray :

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
bytes.writeUTFBytes("Hello World!");
trace("bytes.position is now: " + bytes.position);// 12
```

De la même manière, une opération de lecture incrémente la propriété position en fonction du nombre d'octets lus.

```
var bytes:ByteArray = new ByteArray();

trace("bytes.position is initially: " + bytes.position); // 0
bytes.writeUTFBytes("Hello World!");
trace("bytes.position is now: " + bytes.position);// 12
bytes.position = 0;
trace("The first 6 bytes are: " + (bytes.readUTFBytes(6)));//Hello
trace("And the next 6 bytes are: " + (bytes.readUTFBytes(6)));// World!
```

Vous noterez qu'il est possible de définir la propriété position sur un emplacement spécifique dans la classe ByteArray afin de lire ou d'écrire selon ce décalage.

Propriétés bytesAvailable et length

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les propriétés `length` et `bytesAvailable` vous indiquent la longueur d'une classe ByteArray et le nombre d'octets restants entre la position active et la fin. L'exemple suivant illustre des modes d'utilisation de ces propriétés. L'exemple écrit une chaîne de texte dans la classe ByteArray, puis lit la classe ByteArray octet par octet jusqu'à la détection du caractère « a » ou l'arrivée au terme (`bytesAvailable <= 0`).

```
var bytes:ByteArray = new ByteArray();
var text:String = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vivamus etc.";

bytes.writeUTFBytes(text); // write the text to the ByteArray
trace("The length of the ByteArray is: " + bytes.length);// 70
bytes.position = 0; // reset position
while (bytes.bytesAvailable > 0 && (bytes.readUTFBytes(1) != 'a')) {
    //read to letter a or end of bytes
}
if (bytes.position < bytes.bytesAvailable) {
    trace("Found the letter a; position is: " + bytes.position); // 23
    trace("and the number of bytes available is: " + bytes.bytesAvailable);// 47
}
```

Propriété endian

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les ordinateurs ne stockent pas tous de la même manière les nombres à plusieurs octets, autrement dit, les nombres nécessitant plus d'un octet de mémoire de stockage. Un nombre entier, par exemple, peut prendre 4 octets (ou 32 bits) de mémoire. Certains ordinateurs commencent par stocker l'octet le plus important du nombre (dans l'adresse mémoire de plus bas niveau) tandis que d'autres stockent d'abord l'octet le moins important. Cet attribut d'ordinateur (ou ordre d'octets) est connu comme étant soit *gros-boutiste* (l'octet le plus important en premier lieu) soit *petit-boutiste* (l'octet le moins important en premier lieu). Par exemple, le nombre 0x31323334 serait stocké de la manière suivante pour les ordres d'octets gros-boutiste et petit-boutiste, où a0 représente l'adresse mémoire de plus bas niveau des 4 octets et a3 correspond à celle de plus haut niveau :

Gros-boutiste	Gros-boutiste	Gros-boutiste	Gros-boutiste
a0	a1	a2	a3
31	32	33	34

Petit-boutiste	Petit-boutiste	Petit-boutiste	Petit-boutiste
a0	a1	a2	a3
34	33	32	31

La propriété `endian` de la classe ByteArray vous permet de désigner cet ordre d'octets pour les nombres à plusieurs octets que vous traitez. Les valeurs admises pour cette propriété sont `bigEndian` (gros-boutiste) et `littleEndian` (petit-boutiste). La classe Endian définit les constantes `BIG_ENDIAN` et `LITTLE_ENDIAN` pour configurer la propriété `endian` à l'aide de ces chaînes.

Méthodes compress() et uncompress()

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `compress()` vous permet de compresser une classe ByteArray conformément à un algorithme de compression spécifié sous forme de paramètre. La méthode `uncompress()` vous permet de décompresser un tableau ByteArray compressé conformément à un algorithme de compression. Une fois que vous avez appelé `compress()` et `uncompress()`, la longueur du tableau d'octets est fixée selon la nouvelle longueur et la propriété position est configurée sur la fin.

La classe CompressionAlgorithm définit des constantes pouvant servir à spécifier l'algorithme de compression. La classe ByteArray prend en charge les algorithmes deflate (AIR uniquement), zlib, et lzma. Le format de données compressé zlib est décrit à l'adresse http://www.ietf.org/rfc/rfc1950.txt. L'algorithme lzma a été ajouté à Flash Player 11.4 et AIR 3.4. Il est décrit sur la page suivante : http://www.7-zip.org/7z.html.

L'algorithme de compression deflate est utilisé dans plusieurs formats de compression tels que zlib, gzip et certaines implémentations zip. L'algorithme de compression deflate est décrit à l'adresse http://www.ietf.org/rfc/rfc1951.txt.

Dans l'exemple suivant, un tableau ByteArray appelé `bytes` est compressé à l'aide de l'algorithme Izma :

```
bytes.compress(CompressionAlgorithm.LZMA);
```

Dans l'exemple suivant, un tableau ByteArray compressé est décompressé à l'aide de l'algorithme deflate :

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

Lecture et écriture d'objets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes `readObject()` et `writeObject()` permettent de lire un objet à partir d'une classe ByteArray et d'en écrire un dans une telle classe. Cet objet est codé au format AMF sérialisé. AMF est un protocole de message propriétaire créé par Adobe et utilisé par différentes classes ActionScript 3.0, notamment les objets Netstream, NetConnection, NetStream, LocalConnection et Shared.

Un marqueur de type « un octet » décrit le type des données codées qui suivent. Le format AMF fait appel aux 13 types de données suivants :

```
value-type = undefined-marker | null-marker | false-marker | true-marker | integer-type |
    double-type | string-type | xml-doc-type | date-type | array-type | object-type |
    xml-type | byte-array-type
```

Les données codées suivent le marqueur de type à moins que ce dernier ne représente qu'une seule valeur possible (nulle, vraie ou fausse, par exemple), auquel cas aucun autre élément n'est codé.

Il existe deux versions du format AMF : AMF0 et AMF3. AMF 0 prend en charge l'envoi d'objets complexes par référence et permet aux points de fin de restaurer les relations entre objets. AMF 3 améliore le format AMF 0 en envoyant les traits et chaînes de l'objet par référence, en plus des références d'objet, et en prenant en charge de nouveaux types de données introduits dans la version 3.0 d'ActionScript. La propriété `ByteArray.objectEcoding` spécifie la version du format AMF utilisée pour coder les données d'objet. La classe flash.net.ObjectEncoding définit des constantes permettant de préciser la version du format AMF : `ObjectEncoding.AMF0` et `ObjectEncoding.AMF3`.

Dans l'exemple suivant, `writeObject()` est appelé pour écrire un objet XML dans une classe ByteArray, qu'il compresse ensuite à l'aide de l'algorithme Deflate et écrit dans le fichier `order` situé dans le poste de travail. Dans cet exemple, une étiquette affiche le message « Wrote order file to desktop! » (Fichier d'ordre écrit dans le poste de travail) dans la fenêtre d'AIR lorsque l'opération est terminée.

```
import flash.filesystem.*;
import flash.display.Sprite;
import flash.display.TextField;
import flash.utils.ByteArray;
public class WriteObjectExample extends Sprite
{
    public function WriteObjectExample()
    {
        var bytes:ByteArray = new ByteArray();
        var myLabel:TextField = new TextField();
        myLabel.x = 150;
        myLabel.y = 150;
        myLabel.width = 200;
        addChild(myLabel);

        var myXML:XML =
            <order>
                <item id='1'>
                    <menuName>burger</menuName>
                    <price>3.95</price>
                </item>
                <item id='2'>
                    <menuName>fries</menuName>
                    <price>1.45</price>
                </item>
            </order>;

        // Write XML object to ByteArray
```

```
        bytes.writeObject(myXML);
        bytes.position = 0;//reset position to beginning
        bytes.compress(CompressionAlgorithm.DEFLATE);// compress ByteArray
        writeBytesToFile("order.xml", bytes);
        myLabel.text = "Wrote order file to desktop!";
    }

    private function writeBytesToFile(fileName:String, data:ByteArray):void
    {
        var outFile:File = File.desktopDirectory; // dest folder is desktop
        outFile = outFile.resolvePath(fileName);  // name of file to write
        var outStream:FileStream = new FileStream();
        // open output file stream in WRITE mode
        outStream.open(outFile, FileMode.WRITE);
        // write out the file
        outStream.writeBytes(data, 0, data.length);
        // close it
        outStream.close();
    }
}
```

La méthode `readObject()` lit un objet au format AMF sérialisé à partir d'une classe ByteArray et le stocke dans un objet du type spécifié. Dans l'exemple suivant, le fichier `order` est lu à partir du poste de travail dans une classe ByteArray (`inBytes`), puis il est décompressé. `readObject()` est ensuite appelé pour stocker le fichier dans l'objet XML `orderXML`. Dans cet exemple, une construction de boucle `for each()` est utilisée pour ajouter chaque nœud à une zone de texte à des fins d'affichage. L'exemple présente également la valeur de la propriété `objectEncoding` accompagnée d'un en-tête pour le contenu du fichier `order`.

```
import flash.filesystem.*;
import flash.display.Sprite;
import flash.display.TextField;
import flash.utils.ByteArray;

public class ReadObjectExample extends Sprite
{
    public function ReadObjectExample()
    {
        var inBytes:ByteArray = new ByteArray();
        // define text area for displaying XML content
        var myTxt:TextField = new TextField();
        myTxt.width = 550;
        myTxt.height = 400;
        addChild(myTxt);
        //display objectEncoding and file heading
        myTxt.text = "Object encoding is: " + inBytes.objectEncoding + "\n\n" + "order file: \n\n";
        readFileIntoByteArray("order", inBytes);

        inBytes.position = 0; // reset position to beginning
        inBytes.uncompress(CompressionAlgorithm.DEFLATE);
        inBytes.position = 0;//reset position to beginning
        // read XML Object
        var orderXML:XML = inBytes.readObject();
```

```
        // for each node in orderXML
        for each (var child:XML in orderXML)
        {
            // append child node to text area
            myTxt.text += child + "\n";
        }
    }

    // read specified file into byte array
    private function readFileIntoByteArray(fileName:String, data:ByteArray):void
    {
        var inFile:File = File.desktopDirectory; // source folder is desktop
        inFile = inFile.resolvePath(fileName);  // name of file to read
        var inStream:FileStream = new FileStream();
        inStream.open(inFile, FileMode.READ);
        inStream.readBytes(data);
        inStream.close();
    }
}
```

Exemple ByteArray : lecture d'un fichier .zip

Adobe AIR 1.0 et les versions ultérieures

Cet exemple indique comment lire un fichier .zip simple contenant plusieurs fichiers de types différents. Pour parvenir à lire un tel fichier, les données pertinentes sont extraites des métadonnées pour chacun des fichiers, lesquels sont décompressés dans des classes ByteArray individuelles et écrits dans le poste de travail.

La structure générale d'un fichier .zip repose sur la spécification PKWARE Inc., laquelle est tenue à jour à l'adresse http://www.pkware.com/documents/casestudies/APPNOTE.TXT. Elle commence par l'en-tête et les données du premier fichier de l'archive .zip, suivis de la paire en-tête/données de chaque fichier supplémentaire. (La structure de l'en-tête des fichiers est décrite plus loin dans ce document.) Le fichier .zip comprend éventuellement un enregistrement du descripteur de données (généralement lorsque le fichier zip de sortie a été créé dans la mémoire au lieu d'être enregistré sur un disque). Divers éléments facultatifs viennent ensuite : en-tête de déchiffrement de l'archive, enregistrement des données supplémentaires de l'archive, structure de répertoires centrale, fin Zip64 de l'enregistrement de répertoires central, fin Zip64 du localisateur de répertoires central et fin de l'enregistrement de répertoires.

Le code présenté dans cet exemple a été rédigé dans le seul objectif d'analyser des fichiers zip ne contenant pas de dossiers ; il n'attend pas d'enregistrements de descripteurs de données. Il ne tient pas compte des informations suivant les données du dernier fichier.

Le format de l'en-tête de fichier de chaque fichier est défini de la manière suivante :

signature de l'en-tête de fichier	4 octets
version requise	2 octets
indicateur de bit à usage général	2 octets
méthode de compression	2 octets (8=DEFLATE; 0=UNCOMPRESSED)
heure de la dernière modification du fichier	2 octets

date de la dernière modification du fichier	2 octets
crc-32	4 octets
taille compressée	4 octets
taille décompressée	4 octets
longueur du nom de fichier	2 octets
longueur du champ supplémentaire	2 octets
nom du fichier	variable
champ supplémentaire	variable

L'en-tête du fichier est suivi des véritables données du fichier, au format compressé ou décompressé, suivant l'indicateur de méthode de compression utilisé. L'indicateur est égal à 0 (zéro) si les données du fichier sont décompressées, à 8 si les données sont compressées à l'aide de l'algorithme DEFLATE ou à une autre valeur lorsque les données utilisent d'autres algorithmes de compression.

L'interface utilisateur choisie dans cet exemple se compose d'une étiquette et d'une zone de texte (`taFiles`). L'application écrit les informations suivantes dans la zone de texte pour chaque fichier détecté dans le fichier .zip : nom du fichier, taille compressée et taille décompressée. Le document MXML suivant définit l'interface utilisateur de la version Flex de l'application :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
creationComplete="init();">
    <mx:Script>
        <![CDATA[
            // The application code goes here
        ]]>
    </mx:Script>
    <mx:Form>
        <mx:FormItem label="Output">
            <mx:TextArea id="taFiles" width="320" height="150"/>
        </mx:FormItem>
    </mx:Form>
</mx:WindowedApplication>
```

Le début du programme effectue les tâches suivantes :

- Importation des classes requises

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- Définition de l'interface utilisateur pour Flash

```
import fl.controls.*;

//requires TextArea and Label components in the Library
var taFiles = new TextArea();
var output = new Label();
taFiles.setSize(320, 150);
taFiles.move(10, 30);
output.move(10, 10);
output.width = 150;
output.text = "Contents of HelloAir.zip";
addChild(taFiles);
addChild(output);
```

- Définition de la classe ByteArray `bytes`

```
var bytes:ByteArray = new ByteArray();
```

- Définition des variables destinées à stocker les métadonnées provenant de l'en-tête du fichier

```
// variables for reading fixed portion of file header
var fileName:String = new String();
var flNameLength:uint;
var xfldLength:uint;
var offset:uint;
var compSize:uint;
var uncompSize:uint;
var compMethod:int;
var signature:int;
```

- Définition des objets File (`zfile`) et FileStream (`zStream`) devant représenter le fichier .zip et indication de l'emplacement du fichier .zip à partir duquel les fichiers sont extraits (fichier intitulé HelloAIR.zip dans le répertoire du poste de travail)

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

Dans Flex, le code du programme commence dans la méthode `init()`, laquelle est appelée en tant que gestionnaire `creationComplete` pour la balise `mx:WindowedApplication` racine.

```
// for Flex
private function init():void
{
```

Le programme commence par ouvrir le fichier .zip en mode READ (lecture).

```
zStream.open(zfile, FileMode.READ);
```

Il configure ensuite la propriété `endian` de `bytes` sur la valeur `LITTLE_ENDIAN` afin d'indiquer que l'ordre d'octets des champs numériques commence par l'octet le moins important.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

Une instruction `while()` commence ensuite une boucle qui s'interrompt uniquement lorsque la position active dans le flux de fichier est supérieure ou égale à la taille du fichier.

```
while (zStream.position < zfile.size)
{
```

La première instruction au sein de la boucle lit les 30 premiers octets du flux de fichier dans la classe ByteArray `bytes`. Les 30 premiers octets constituent la partie à taille fixe de l'en-tête du premier fichier.

```
// read fixed metadata portion of local file header
zStream.readBytes(bytes, 0, 30);
```

Le code lit ensuite un nombre entier (`signature`) à partir des premiers octets de l'en-tête de 30 octets. La définition du format ZIP indique que la signature de chaque en-tête de fichier correspond à la valeur hexadécimale `0x04034b50` ; si la signature est différente, cela signifie que le code a dépassé la section des fichiers contenus dans le fichier .zip et qu'il ne reste plus aucun fichier à extraire. Dans ce cas, le code quitte immédiatement la boucle `while` au lieu d'attendre la fin du tableau d'octets.

```
bytes.position = 0;
signature = bytes.readInt();
// if no longer reading data files, quit
if (signature != 0x04034b50)
{
    break;
}
```

La partie suivante du code lit l'octet d'en-tête à la position décalée 8 et stocke la valeur dans la variable `compMethod`. Cet octet contient une valeur indiquant la méthode de compression appliquée à ce fichier. Plusieurs méthodes de compression sont autorisées, mais en pratique, presque tous les fichiers .zip utilisent l'algorithme de compression DEFLATE. Si le fichier actif est compressé à l'aide de la méthode de compression DEFLATE, `compMethod` est égal à 8 ; si le fichier n'est pas compressé, `compMethod` est égal à 0.

```
bytes.position = 8;
compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

Les 30 premiers octets sont suivis par une partie d'en-tête à longueur variable contenant le nom du fichier et, éventuellement, un champ supplémentaire. La variable `offset` stocke la taille de cette partie. La taille est calculée en ajoutant la longueur du nom de fichier à la longueur du champ supplémentaire, lue à partir de l'en-tête aux décalages 26 et 28.

```
offset = 0;// stores length of variable portion of metadata
bytes.position = 26;  // offset to file name length
flNameLength = bytes.readShort();// store file name
offset += flNameLength; // add length of file name
bytes.position = 28;// offset to extra field length
xfldLength = bytes.readShort();
offset += xfldLength;// add length of extra field
```

Le programme lit ensuite la partie à longueur variable de l'en-tête du fichier pour identifier le nombre d'octets stockés dans la variable `offset`.

```
// read variable length bytes between fixed-length header and compressed file data
zStream.readBytes(bytes, 30, offset);
```

Le programme lit le nom du fichier à partir de la partie variable de l'en-tête et l'affiche dans la zone de texte accompagné des tailles compressée (zippée) et décompressée (initiale) du fichier.

```
// Flash version
bytes.position = 30;
fileName = bytes.readUTFBytes(flNameLength); // read file name
taFiles.appendText(fileName + "\n"); // write file name to text area
bytes.position = 18;
compSize = bytes.readUnsignedInt();  // store size of compressed portion
taFiles.appendText("\tCompressed size is: " + compSize + '\n');
bytes.position = 22;  // offset to uncompressed size
uncompSize = bytes.readUnsignedInt();  // store uncompressed size
taFiles.appendText("\tUncompressed size is: " + uncompSize + '\n');
```

```
// Flex version
        bytes.position = 30;
        fileName = bytes.readUTFBytes(flNameLength); // read file name
        taFiles.text += fileName + "\n"; // write file name to text area
        bytes.position = 18;
        compSize = bytes.readUnsignedInt();  // store size of compressed portion
        taFiles.text += "\tCompressed size is: " + compSize + '\n';
        bytes.position = 22;  // offset to uncompressed size
        uncompSize = bytes.readUnsignedInt();  // store uncompressed size
        taFiles.text += "\tUncompressed size is: " + uncompSize + '\n';
```

L'exemple de code lit le reste du fichier à partir du flux de fichier en octets (`bytes`) selon la longueur indiquée par la taille compressée, écrasant ainsi l'en-tête de fichier dans les 30 premiers octets. La taille compressée est exacte et ce, même si le fichier n'est pas compressé, car elle équivaut alors à la taille décompressée du fichier.

```
    // read compressed file to offset 0 of bytes; for uncompressed files
    // the compressed and uncompressed size is the same
    if (compSize == 0) continue;
    zStream.readBytes(bytes, 0, compSize);
```

Dans la suite de l'exemple, le fichier compressé est décompressé et la fonction `outfile()` est appelée afin de l'écrire dans le flux du fichier de sortie. Le code passe à `outfile()` le nom du fichier et le tableau d'octets contenant les données du fichier.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

Dans l'exemple précédemment mentionné, `bytes.uncompress(CompressionAlgorithm.DEFLATE)` fonctionne uniquement dans les applications AIR. Pour décompresser les données compressées dans AIR et Flash Player, appelez la fonction `inflate()` de ByteArray.

Les accolades de fermeture indiquent la fin de la boucle `while`, de la méthode `init()` et du code de l'application Flex, à l'exception de la méthode `outFile()`. L'exécution revient au début de la boucle `while` et poursuit le traitement des octets suivants du fichier .zip, soit en extrayant un autre fichier soit en mettant un terme au traitement du fichier .zip si le dernier fichier a été traité.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

La fonction `outfile()` ouvre un fichier de sortie en mode WRITE (écriture) dans le poste de travail en lui donnant le nom fourni par le paramètre `filename`. Elle écrit ensuite les données du fichier issues du paramètre `data` dans le flux du fichier de sortie (`outStream`) et ferme le fichier.

```
// Flash version
function outFile(fileName:String, data:ByteArray):void
{
    var outFile:File = File.desktopDirectory; // destination folder is desktop
    outFile = outFile.resolvePath(fileName);  // name of file to write
    var outStream:FileStream = new FileStream();
    // open output file stream in WRITE mode
    outStream.open(outFile, FileMode.WRITE);
    // write out the file
    outStream.writeBytes(data, 0, data.length);
    // close it
    outStream.close();
}

private function outFile(fileName:String, data:ByteArray):void
{
    var outFile:File = File.desktopDirectory; // dest folder is desktop
    outFile = outFile.resolvePath(fileName);  // name of file to write
    var outStream:FileStream = new FileStream();
    // open output file stream in WRITE mode
    outStream.open(outFile, FileMode.WRITE);
    // write out the file
    outStream.writeBytes(data, 0, data.length);
    // close it
    outStream.close();
}
```

Chapitre 42 : Principes de base de la mise en réseau et de la communication

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez des applications dans Flash Player ou AIR, il est souvent nécessaire d'accéder à des ressources externes. Vous pouvez, par exemple, demander une image à un serveur Web Internet et obtenir en retour les données correspondantes. Vous pouvez également échanger des objets sérialisés avec un serveur d'applications via une connexion socket. Les API de Flash Player et d'AIR proposent plusieurs classes qui permettent aux applications de participer à cet échange. Elles prennent en charge la mise en réseau IP pour les protocoles tels qu'UDP, TCP, HTTP, RTMP et RTMFP.

Les classes suivantes permettent d'envoyer et de recevoir des données via un réseau :

Classe	Formats de données pris en charge	Protocoles	Description
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	Charge les types de données pris en charge et convertit les données en objet d'affichage. Voir « Chargement dynamique du contenu d'affichage » à la page 205.
URLLoader	Tous les formats (texte, XML, binaire, etc.)	HTTP, HTTPS	Charge les formats de données arbitraires. L'application est responsable de l'interprétation des données. Voir « Utilisation de la classe URLLoader » à la page 847
FileReference	Tous les formats	HTTP	Charge et télécharge les fichiers. Voir « Utilisation de la classe FileReference » à la page 676
NetConnection	Vidéo, audio, AMF (ActionScript Message Format)	HTTP, HTTPS, RTMP, RTMFP	Etablit une connexion aux flux vidéo, audio et d'objets distants. Voir « Utilisation de la vidéo » à la page 489.
Sound	Audio	HTTP	Charge et lit les formats audio pris en charge. Voir « Chargement de fichiers audio externes » à la page 458
XMLSocket	XML	TCP	Echange des messages XML avec un serveur XMLSocket. Voir « Sockets XML » à la page 834.
Socket	Tous les formats	TCP	Etablit une connexion à un serveur socket TCP. Voir « Sockets clients binaires » à la page 829.
SecureSocket (AIR)	Tous les formats	TCP avec SSLv3 ou TLSv1	Etablit une connexion à un serveur socket TCP qui requiert une sécurité SSL ou TLS. Voir « Sockets de client sécurisés (AIR) » à la page 829
ServerSocket (AIR)	Tous les formats	TCP	Sert de serveur aux connexions socket TCP entrantes. Voir « Sockets de serveur » à la page 837.
DatagramSocket (AIR)	Tous les formats	UDP	Envoie et reçoit des paquets UDP. Voir « Sockets UDP (AIR) » à la page 839

Lors de la création d'une application Web, il s'avère souvent utile de stocker les informations persistantes relatives à l'état de l'application de l'utilisateur. Les applications et pages HTML utilisent généralement des cookies à cet effet. Dans Flash Player, l'utilisation de la classe SharedObject mène à un résultat identique. Voir « Objets partagés » à la page 728. (Vous pouvez utiliser la classe SharedObject dans les applications AIR, mais se contenter d'enregistrer les données dans un fichier standard donne lieu à moins de restrictions.)

Lorsque l'application Flash Player ou AIR doit communiquer avec une autre application Flash Player ou AIR sur le même ordinateur, vous pouvez faire appel à la classe LocalConnection. Deux processus SWF ou plus hébergés sur la même page Web peuvent ainsi communiquer entre eux. Un processus SWF qui s'exécute sur une page Web peut de même communiquer avec une application AIR. Voir « Communications avec d'autres occurrences de Flash Player et d'AIR » à la page 862

Pour communiquer avec d'autres processus non-SWF sur l'ordinateur local, vous disposez de la classe NativeProcess intégrée à AIR 2. La classe NativeProcess permet à l'application AIR de démarrer et de communiquer avec d'autres applications. Voir « Communication avec les processus natifs dans AIR » à la page 869

Les classes suivantes permettent d'obtenir des informations sur l'environnement réseau de l'ordinateur sur lequel s'exécute une application AIR :

- NetworkInfo : fournit des informations sur les interfaces réseau disponibles, telles que l'adresse IP de l'ordinateur. Voir « Interfaces réseau » à la page 822.
- DNSResolver : permet de consulter les enregistrements DNS. Voir « Enregistrements DNS (Domain Name System) » à la page 826
- ServiceMonitor : permet de surveiller la disponibilité d'un serveur. Voir « Surveillance des services » à la page 824.
- URLMonitor : permet de surveiller la disponibilité d'une ressource associée à une URL donnée. Voir « Surveillance HTTP » à la page 825.
- SocketMonitor et SecureSocketMonitor : permettent de surveiller la disponibilité d'une ressource sur un socket. Voir « Surveillance des sockets » à la page 825.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à la programmation de code de mise en réseau et de communication :

Données externes Données stockées sous une certaine forme en dehors de l'application et chargées dans cette dernière si besoin est. Ces données peuvent être stockées dans un fichier chargé directement, dans une base de données ou sous une autre forme récupérée en appelant des scripts ou des programmes exécutés sur un serveur.

Variables codées URL Le format codé URL permet de représenter plusieurs variables (paires de valeurs et de noms de variable) dans une seule chaîne de texte. Les variables individuelles sont écrites dans le format nom=valeur. Chaque variable (c'est-à-dire, chaque paire nom-valeur) est séparée par un caractère esperluette, comme suit : variable1=valeur1&variable2=valeur2. De cette façon, un nombre infini de variables peut être envoyé sous la forme d'un seul message.

Type MIME Code standard utilisé pour identifier le type d'un fichier donné dans une communication Internet. Tous les types de fichier ont un code spécifique utilisé pour les identifier. Lors de l'envoi d'un fichier ou d'un message, un ordinateur (un serveur Web ou l'occurrence Flash Player ou AIR d'un utilisateur, par exemple) spécifie le type de fichier envoyé.

HTTP Hypertext Transfer Protocol : format standard de livraison de pages Web et de différents types de contenu envoyés sur Internet.

Méthode de requête Lorsqu'une application (tel un navigateur Web ou une application AIR) envoie un message (appelé requête HTTP) à un serveur Web, les données envoyées peuvent être intégrées à la requête de deux façons

différentes : les méthodes de requête GET et POST. Côté serveur, le programme qui reçoit la requête doit pouvoir rechercher les données dans la portion appropriée de la requête. C'est pourquoi la méthode de requête utilisée pour envoyer des données de votre application doit correspondre à celle utilisée pour lire ces données sur le serveur.

Connexions socket Connexion persistante pour la communication entre deux ordinateurs.

Charger Envoyer un fichier à un autre ordinateur.

Télécharger Récupérer un fichier d'un autre ordinateur.

Interfaces réseau

Adobe AIR 2 et ultérieur

L'objet NetworkInfo permet de vérifier les interfaces matérielles et logicielles réseau dont dispose l'application. Puisqu'il s'agit d'un objet *singleton*, il est inutile de le créer. Accédez à l'unique objet NetworkInfo par le biais de la propriété de classe statique `networkInfo`. L'objet NetworkInfo distribue également un événement `networkChange` en cas de modification de l'une des interfaces disponibles.

Appelez la méthode `findInterfaces()` pour obtenir la liste des objets NetworkInterface. Chaque objet NetworkInterface de la liste décrit l'une des interfaces disponibles. L'objet NetworkInterface fournit des informations telles que l'adresse IP, l'adresse matérielle, l'unité maximale de transmission et indique si l'interface est active.

L'exemple de code suivant effectue le suivi des propriétés NetworkInterface de chaque interface sur l'ordinateur client :

```
package {
import flash.display.Sprite;
import flash.net.InterfaceAddress;
import flash.net.NetworkInfo;
import flash.net.NetworkInterface;

public class NetworkInformationExample extends Sprite
{
    public function NetworkInformationExample()
    {
        var networkInfo:NetworkInfo = NetworkInfo.networkInfo;
        var interfaces:Vector.<NetworkInterface> = networkInfo.findInterfaces();

        if( interfaces != null )
        {
            trace( "Interface count: " + interfaces.length );
            for each ( var interfaceObj:NetworkInterface in interfaces )
            {
                trace( "\nname: " + interfaceObj.name );
                trace( "display name: " + interfaceObj.displayName );
                trace( "mtu: "  + interfaceObj.mtu );
```

```
                trace( "active?: " + interfaceObj.active );
                trace( "parent interface: " + interfaceObj.parent );
                trace( "hardware address: " + interfaceObj.hardwareAddress );
                if( interfaceObj.subInterfaces != null )
                {
                    trace( "# subinterfaces: " + interfaceObj.subInterfaces.length );
                }
                trace("# addresses: " + interfaceObj.addresses.length );
                for each ( var address:InterfaceAddress in interfaceObj.addresses )
                {
                    trace( "  type: "           + address.ipVersion );
                    trace( "  address: " + address.address );
                    trace( "  broadcast: " + address.broadcast );
                    trace( "  prefix length: " + address.prefixLength );
                }
            }
        }
    }
}
}
```

Pour plus d'informations, voir :

- NetworkInfo
- NetworkInterface
- InterfaceAddress
- Flexpert: Detecting the network connection type with Flex 4.5 (disponible en anglais uniquement)

Changements de connectivité réseau

Adobe AIR 1.0 et les versions ultérieures

Votre application AIR peut s'exécuter dans des environnements caractérisés par une connectivité réseau non déterminée et en évolution constante. Pour aider une application à gérer les connexions aux ressources en ligne, Adobe AIR envoie un événement de changement réseau à chaque fois qu'une connexion réseau est disponible ou non disponible. L'objet NetworkInfo et l'objet NativeApplication de l'application distribuent tous deux l'événement `networkChange`. Pour répondre à cet événement, ajoutez un écouteur :

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

Définissez également une fonction de gestionnaire d'événement :

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

L'événement `networkChange` n'indique pas un changement de l'activité réseau globale. Il se contente de signaler qu'une connexion réseau donnée a changé. AIR ne tente pas d'interpréter la signification du changement réseau. Un ordinateur connecté au réseau étant susceptible de disposer d'un nombre élevé de connexions réelles et virtuelles, perdre une connexion ne signifie pas nécessairement perdre une ressource. A l'inverse, une nouvelle connexion n'est pas non plus synonyme de disponibilité accrue des ressources. Il arrive qu'une nouvelle connexion bloque l'accès à des ressources précédemment disponibles (lors d'une connexion à un VPN, par exemple).

En règle générale, l'unique façon de déterminer si une application peut se connecter à une ressource distante consiste à en faire l'essai. La structure de surveillance des services propose une technique orientée événement de réponse aux changements de connectivité réseau vers un hôte déterminé.

Remarque : *la structure de surveillance des services détecte si un serveur répond de manière acceptable à une requête. Le succès du contrôle n'est pas synonyme de connectivité totale. Les services Web évolutifs proposent souvent des dispositifs de mise en cache et d'équilibrage de charge destinés à réorienter le trafic vers un groupe de serveurs Web. Dans ce cas de figure, les prestataires de services n'assurent qu'un diagnostic partiel de la connectivité réseau.*

Surveillance des services

Adobe AIR 1.0 et les versions ultérieures

La structure de surveillance des services, distincte de la structure AIR, réside dans le fichier aircore.swc. Pour pouvoir utiliser la structure, le fichier aircore.swc doit être inclus dans le processus de compilation.

Adobe® Flash® Builder inclut automatiquement cette bibliothèque.

La classe ServiceMonitor met en œuvre la structure de surveillance des services réseau et propose des fonctionnalités de base aux utilitaires de surveillance des services. Par défaut, une occurrence de la classe ServiceMonitor distribue des événements relatifs à la connectivité réseau. L'objet ServiceMonitor distribue ces événements lors de la création de l'occurrence et lorsque le moteur d'exécution détecte un changement au sein du réseau. Par ailleurs, vous pouvez définir la propriété `pollInterval` d'une occurrence de ServiceMonitor de sorte à vérifier la connectivité à fréquence déterminée exprimée en millisecondes, sans tenir compte des événements de connectivité réseau généraux. Un objet ServiceMonitor ne vérifie pas la connectivité réseau tant que la méthode `start()` n'a pas été appelée.

La classe URLMonitor, une sous-classe de la classe ServiceMonitor, détecte les changements de connectivité HTTP associés à une requête URLRequest déterminée.

La classe SocketMonitor, également une sous-classe de la classe ServiceMonitor, détecte les changements de connectivité vers un hôte déterminé sur un port donné.

Remarque : *dans les versions antérieures à AIR 2, la structure de surveillance des services était publiée dans la bibliothèque servicemonitor.swc. L'utilisation de cette bibliothèque est à présent déconseillée. Utilisez plutôt la bibliothèque aircore.swc.*

Flash CS4 et CS5 Professional

Pour utiliser ces classes dans Adobe® Flash® CS4 ou CS5 Professional :

1. Choisissez Fichier > Paramètres de publication.
2. Cliquez sur le bouton Paramètres pour ActionScript 3.0. Sélectionnez Chemin de la bibliothèque.
3. Cliquez sur le bouton Localiser le fichier SWC et accédez au dossier AIK dans le dossier d'installation de Flash Professional.
4. Dans ce dossier, recherchez le fichier /frameworks/libs/air/aircore.swc (AIR 2) ou /frameworks/libs/air/servicemonitor.swc (AIR 1.5).

5 Cliquez sur le bouton OK.

6 Ajoutez l'instruction d'importation suivante dans votre code ActionScript 3.0 :

```
import air.net.*;
```

Flash CS3 Professional

Pour utiliser ces classes dans Adobe® Flash® CS3 Professional, faites glisser le composant ServiceMonitorShim du panneau Composants vers la bibliothèque. Ajoutez ensuite l'instruction `import` suivante au code ActionScript 3.0 :

```
import air.net.*;
```

Surveillance HTTP

Adobe AIR 1.0 et les versions ultérieures

La classe URLMonitor détermine s'il est possible d'envoyer des requêtes HTTP à une adresse déterminée sur le port 80 (qui est généralement utilisé par les communications HTTP). Le code suivant utilise une occurrence de la classe URLMonitor pour détecter les changements de connectivité vers le site Web d'Adobe :

```
import air.net.URLMonitor;
import flash.net.URLRequest;
import flash.events.StatusEvent;
var monitor:URLMonitor;
monitor = new URLMonitor(new URLRequest('http://www.example.com'));
monitor.addEventListener(StatusEvent.STATUS, announceStatus);
monitor.start();
function announceStatus(e:StatusEvent):void {
    trace("Status change. Current status: " + monitor.available);
}
```

Surveillance des sockets

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR peuvent également utiliser les connexions de socket pour assurer la connectivité de type « push ». Pour des raisons de sécurité, les pare-feu et routeurs réseau interdisent généralement les communications réseau sur les ports non autorisés. Les développeurs doivent donc tenir compte du fait que les utilisateurs ne sont pas toujours à même d'établir des connexions de socket.

Le code suivant utilise une occurrence de la classe SocketMonitor pour détecter les changements de connectivité vers une connexion de socket : Le port surveillé, 6667, est un port courant pour IRC :

```
import air.net.ServiceMonitor;
import flash.events.StatusEvent;

socketMonitor = new SocketMonitor('www.example.com',6667);
socketMonitor.addEventListener(StatusEvent.STATUS, socketStatusChange);
socketMonitor.start();

function announceStatus(e:StatusEvent):void {
    trace("Status change. Current status: " + socketMonitor.available);
}
```

Si le serveur socket requiert une connexion sécurisée, utilisez la classe SecureSocketMonitor au lieu de SocketMonitor.

Enregistrements DNS (Domain Name System)

Adobe AIR 2.0 et les versions ultérieures

La classe DNSResolver permet de consulter les enregistrements de ressource DNS. Les enregistrements de ressource DNS fournissent des informations telles que l'adresse IP d'un nom de domaine et le nom de domaine d'une adresse IP. Vous pouvez consulter les types suivants d'enregistrements de ressources DNS :

- ARecord : adresse IPv4 d'un hôte
- AAAARecord : adresse IPv6 d'un hôte
- MXRecord : enregistrement d'échange de courrier associé à un hôte
- PTRRecord : nom d'hôte associé à une adresse IP
- SRVRecord : enregistrement associé à un service.

Pour consulter un enregistrement, transmettez une chaîne de requête et l'objet de classe représentant le type d'enregistrement à la méthode `lookup()` de l'objet DNSResolver. La chaîne de requête à utiliser varie selon le type d'enregistrement :

Classe d'enregistrement	Chaîne de requête	Exemple de chaîne de requête
ARecord	nom d'hôte	« example.com »
AAAARecord	nom d'hôte	« example.com »
MXRecord	nom d'hôte	« example.com »
PTRRecord	Adresse IP	« 208.77.188.166 »
SRVRecord	Identifiant de service : _service._protocole.hôte	« _sip._tcp.example.com »

L'exemple de code suivant recherche l'adresse IP de l'hôte « example.com ».

```
package
{
    import flash.display.Sprite;
    import flash.events.DNSResolverEvent;
    import flash.events.ErrorEvent;
    import flash.net.dns.ARecord;
    import flash.net.dns.DNSResolver;

    public class DNSResolverExample extends Sprite
    {

        public function DNSResolverExample()
        {
            var resolver:DNSResolver = new DNSResolver();
            resolver.addEventListener( DNSResolverEvent.LOOKUP, lookupComplete );
            resolver.addEventListener( ErrorEvent.ERROR, lookupError );

            resolver.lookup( "example.com.", ARecord );
        }

        private function lookupComplete( event:DNSResolverEvent ):void
        {
            trace( "Query string: " + event.host );
            trace( "Record count: " + event.resourceRecords.length );
            for each( var record:* in event.resourceRecords )
            {
                if( record is ARecord ) trace( record.address );
            }

        }

        private function lookupError( error:ErrorEvent ):void
        {
            trace("Error: " + error.text );
        }
    }
}
```

Pour plus d'informations, voir :

- DNSResolver
- DNSResolverEvent
- ARecord
- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

Chapitre 43 : Sockets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un socket est un type de connexion réseau établie entre deux processus informatiques. En règle générale, les processus s'exécutent sur deux ordinateurs distincts connectés au même réseau IP (Internet Protocol). Les processus connectés peuvent toutefois s'exécuter sur le même ordinateur via l'adresse IP spéciale « hôte local ».

Adobe Flash Player prend en charge les sockets TCP (Transport Control Protocol) côté client. Une application Flash Player peut se connecter à un autre processus qui fait office de serveur socket, mais est incapable d'accepter les demandes de connexion entrantes provenant d'autres processus. En d'autres termes, une application Flash Player peut se connecter à un serveur TCP, mais ne peut pas faire office de serveur TCP.

L'API de Flash Player comprend également la classe XMLSocket. La classe XMLSocket utilise un protocole propre à Flash Player, qui permet d'échanger des messages XML avec un serveur qui comprend ce protocole. La classe XMLSocket a été introduite dans ActionScript 1 et continue à être prise en charge à des fins de compatibilité ascendante. En règle générale, réservez la classe Socket aux nouvelles applications, sauf si vous vous connectez à un serveur dédié aux communications avec Flash XMLSockets.

Adobe AIR intègre plusieurs autres classes réservées à la programmation réseau basée sur les sockets. Les applications AIR peuvent agir comme serveurs sockets TCP grâce à la classe ServerSocket, et peuvent se connecter à des serveurs sockets qui requièrent une sécurité SSL ou TLS grâce à la classe SecureSocket. La classe DatagramSocket permet également aux applications AIR d'envoyer et de recevoir des messages UDP (Universal Datagram Protocol).

Voir aussi

Package flash.net

« Connexion aux sockets » à la page 1114

Sockets TCP

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le protocole TCP (Transmission Control Protocol) permet d'échanger des messages via une connexion réseau permanente. Le protocole TCP assure la réception dans l'ordre correct de tous les messages envoyés (ce qui élimine les principaux problèmes liés au réseau). Les connexions TCP nécessitent un « client » et un « serveur ». Flash Player peut créer des sockets clients. Adobe AIR peut en outre créer des sockets serveur.

Les API ActionScript suivantes assurent des connexions TCP :

- Socket : permet à une application cliente de se connecter à un serveur. La classe Socket peut écouter les connexions entrantes.
- SecureSocket (AIR) : permet à une application cliente de se connecter à un serveur approuvé et d'établir des communications chiffrées.
- ServerSocket (AIR) : permet à une application d'écouter les connexions entrantes et de servir de serveur.
- XMLSocket : permet à une application cliente de se connecter à un serveur XMLSocket.

Sockets clients binaires

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une connexion socket binaire est semblable à un socket XML, à une différence près : le client et le serveur n'échangent pas uniquement des messages XML. La connexion peut en effet transférer des données au format binaire. Vous pouvez ainsi vous connecter à un plus large éventail de services, notamment des serveurs de messagerie (POP3, SMTP et IMAP) et d'informations (NNTP).

Socket, classe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Socket vous permet d'établir des connexions socket, ainsi que de lire et d'écrire des données binaires brutes. La classe Socket est utile si vous utilisez des serveurs faisant appel à des protocoles binaires. Grâce aux connexions socket binaires, vous pouvez écrire du code qui gère les interactions avec différents protocoles Internet (POP3, SMTP, IMAP et NNTP, par exemple). Ces interactions permettent alors aux applications de se connecter à des serveurs de messagerie et d'informations.

Flash Player peut communiquer directement avec un serveur en utilisant directement le protocole binaire de ce serveur. Certains serveurs utilisent l'ordre d'octets « gros-boutiste », d'autres l'ordre « petit-boutiste ». La plupart des serveurs sur Internet utilisent l'ordre gros-boutiste, car il s'agit de l'ordre d'octets du réseau. L'ordre petit-boutiste s'est répandu en raison de son utilisation par l'architecture Intel® x86. Vous devez utiliser l'ordre d'octets correspondant au serveur qui envoie et reçoit les données. Toutes les opérations sont effectuées par les interfaces IDataInput et IDataOutput, et les classes qui les implémentent (ByteArray, Socket et URLStream) sont par défaut encodées au format gros-boutiste (l'octet le plus significatif en premier). Cet ordre d'octet par défaut a été sélectionné pour correspondre à Java et à l'ordre des octets réseau officiel. Pour modifier l'ordre d'octets à utiliser, vous pouvez définir la propriété `endian` sur `Endian.BIG_ENDIAN` ou `Endian.LITTLE_ENDIAN`.

la classe Socket hérite de toutes les méthodes définies par les interfaces IDataInput et IDataOutput (qui résident dans le package flash.utils). Vous devez impérativement utiliser ces méthodes pour écrire et lire à partir du socket.

Pour plus d'informations, voir :

- Socket
- IDataInput
- IDataOutput
- Evénement socketData

Sockets de client sécurisés (AIR)

Adobe AIR 2 et ultérieur

La classe SecureSocket permet d'établir une connexion aux serveurs socket qui utilisent Secure Sockets Layer version 4 (SSLv4) ou Transport Layer Security version 1 (TLSv1). Les avantages que présente un socket sécurisé sont au nombre de trois : authentification du serveur, intégrité des données et confidentialité des messages. Le moteur d'exécution authentifie un serveur par le biais de son certificat et de sa relation aux certificats délivrés par les autorités de certification racine ou intermédiaires dans le magasin d'approbation de l'utilisateur. Le moteur d'exécution se fonde sur les algorithmes de cryptographie utilisés par les implémentations de protocoles SSL et TLS pour assurer l'intégrité des données et la confidentialité des messages.

Lorsque vous vous connectez à un serveur par le biais de l'objet SecureSocket, le moteur d'exécution valide le certificat du serveur via le magasin d'approbation des certificats. Sous Windows et Mac, le système d'exploitation fournit le magasin d'approbation. Sous Linux, le moteur d'exécution fournit son propre magasin d'approbation.

Si le certificat du serveur n'est ni valide, ni fiable, le moteur d'exécution distribue un événement `ioError`. La propriété `serverCertificateStatus` de l'objet SecureSocket permet d'identifier le motif de l'échec de la validation. Il est impossible de communiquer avec un serveur qui ne dispose pas d'un certificat valide et fiable.

La classe CertificateStatus définit les constantes de chaîne qui représentent les résultats possibles de la validation :

- Expired (Expiré) : la date d'expiration du certificat est dépassée.
- Invalid (Non valide) : diverses raisons expliquent la non-validité d'un certificat. Celui-ci pourrait avoir été modifié, altéré ou son type pourrait être incorrect.
- Invalid chain (Chaîne non valide) : un ou plusieurs certificats de la chaîne de certificats du serveur ne sont pas valides.
- Principal mismatch (Non-concordance majeure) : le nom d'hôte du serveur et le nom commun figurant sur le certificat ne sont pas identiques. En d'autres termes, le serveur utilise le certificat d'un autre serveur.
- Revoked (Révoqué) : l'autorité de certification a révoqué le certificat.
- Trusted (Fiable) : le certificat est valide et fiable. Pour qu'un objet SecureSocket puisse se connecter à un serveur, ce dernier doit disposer d'un certificat valide et fiable.
- Unknown (Inconnu) : l'objet SecureSocket n'a pas encore validé le certificat. La propriété `serverCertificateStatus` possède cet état avant que vous n'appeliez `connect()` et avant la distribution d'un événement `connect` ou `ioError`.
- Untrusted signers (Signataires non fiables) : le certificat ne fait pas partie d'une « chaîne » de certificats racine fiables dans le magasin d'approbation de l'ordinateur client.

L'établissement de communications avec un objet SecureSocket nécessite un serveur qui utilise un protocole sécurisé et possède un certificat valide et fiable. A d'autres égards, l'utilisation d'un objet SecureSocket s'apparente à celle d'un objet Socket.

L'objet SecureSocket n'est pas pris en charge par toutes les plates-formes. Faites appel à la propriété `isSupported` de la classe SecureSocket pour vérifier si le moteur d'exécution gère l'utilisation de l'objet SecureSocket sur l'ordinateur client actif.

Pour plus d'informations, voir :

- SecureSocket
- CertificateStatus
- IDataInput
- IDataOutput
- Evénement socketData

Exemple de socket TCP : création d'un client Telnet

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'exemple Telnet illustre les techniques de connexion à un serveur distant et de transmission des données à l'aide de la classe Socket. Cet exemple étudie les techniques suivantes :

- Création d'un client Telnet personnalisé à l'aide de la classe Socket

- Envoi de texte au serveur distant à l'aide de l'objet ByteArray
- Gestion des données reçues d'un serveur distant

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application Telnet se trouvent dans le dossier Samples/Telnet. L'application se compose des fichiers suivants :

Fichier	Description
TelnetSocket.fla ou TelnetSocket.mxml	Fichier d'application principal correspondant à l'interface utilisateur de Flex (MXML) ou Flash (FLA).
TelnetSocket.as	Classe Document fournissant la logique de l'interface utilisateur (Flash uniquement).
com/example/programmingas3/Telnet/Telnet.as	Fournit la fonctionnalité client Telnet à l'application, par exemple pour la connexion à un serveur distant et l'envoi, la réception et l'affichage des données.

Présentation de l'application socket Telnet

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le fichier principal TelnetSocket.mxml se charge de créer l'interface utilisateur (IU) de l'application entière.

Outre l'IU, ce fichier définit deux méthodes, `login()` et `sendCommand()`, qui permettent la connexion de l'utilisateur au serveur spécifié.

L'exemple suivant répertorie le code ActionScript du fichier d'application principal :

```
import com.example.programmingas3.socket.Telnet;

private var telnetClient:Telnet;
private function connect():void
{
    telnetClient = new Telnet(serverName.text, int(portNumber.text), output);
    console.title = "Connecting to " + serverName.text + ":" + portNumber.text;
    console.enabled = true;
}
private function sendCommand():void
{
    var ba:ByteArray = new ByteArray();
    ba.writeMultiByte(command.text + "\n", "UTF-8");
    telnetClient.writeBytesToSocket(ba);
    command.text = "";
}
```

La première ligne de code importe la classe Telnet à partir du package personnalisé com.example.programmingas.socket. La deuxième ligne de code déclare une occurrence de la classe Telnet, `telnetClient`, qui est initialisée ultérieurement par la méthode `connect()`. Ensuite, la méthode `connect()` est déclarée et initialise la variable `telnetClient` déclarée auparavant. Cette méthode transmet le nom du serveur Telnet spécifié par l'utilisateur, le port de ce serveur et une référence à un composant TextArea dans la liste d'affichage, qui sert à afficher les réponses textuelles reçues du serveur socket. Les deux dernières lignes de la méthode `connect()` définissent la propriété `title` du composant Panel et active celui-ci, ce qui permet à l'utilisateur d'envoyer les données au serveur distant. La méthode finale du fichier d'application principal, `sendCommand()`, permet d'envoyer les commandes de l'utilisateur au serveur distant sous forme d'objet ByteArray.

Présentation de la classe Telnet

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Telnet se charge d'établir la connexion au serveur distant Telnet et d'envoyer et de recevoir les données.

La classe Telnet déclare les variables privées suivantes :

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

La première variable, `serverURL`, contient l'adresse du serveur auquel se connecter, spécifiée par l'utilisateur.

La deuxième variable, `portNumber`, correspond au numéro de port sur lequel le serveur Telnet s'exécute actuellement. Par défaut, le service Telnet utilise le port 23.

La troisième variable, `socket`, est une occurrence de Socket qui tente d'établir une connexion au serveur défini par les variables `serverURL` et `portNumber`.

La quatrième variable, `ta`, est une référence à l'occurrence du composant TextArea sur la scène. Ce composant sert à afficher les réponses provenant du serveur Telnet distant ou les éventuels messages d'erreur.

La dernière variable, `state`, est une valeur numérique utilisée pour déterminer les options prises en charge par le client Telnet.

Comme vous l'avez vu précédemment, la fonction constructeur de la classe Telnet est appelée par la méthode `connect()` dans le fichier d'application principal.

Le constructeur Telnet prend trois paramètres : `server`, `port` et `output`. Les paramètres `server` et `port` spécifient le nom et le numéro de port du serveur Telnet. Le dernier paramètre, `output`, est une référence à une occurrence du composant TextArea sur la scène où s'affichent les résultats du serveur à l'intention de l'utilisateur.

```
public function Telnet(server:String, port:int, output:TextArea)
{
    serverURL = server;
    portNumber = port;
    ta = output;
    socket = new Socket();
    socket.addEventListener(Event.CONNECT, connectHandler);
    socket.addEventListener(Event.CLOSE, closeHandler);
    socket.addEventListener(ErrorEvent.ERROR, errorHandler);
    socket.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);
    socket.addEventListener(ProgressEvent.SOCKET_DATA, dataHandler);
    Security.loadPolicyFile("http://" + serverURL + "/crossdomain.xml");
    try
    {
        msg("Trying to connect to " + serverURL + ":" + portNumber + "\n");
        socket.connect(serverURL, portNumber);
    }
    catch (error:Error)
    {
        msg(error.message + "\n");
        socket.close();
    }
}
```

Ecriture de données dans un socket

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour écrire des données sur une connexion socket, vous appelez l'une des méthodes d'écriture de la classe Socket. Parmi ces méthodes d'écriture figurent `writeBoolean()`, `writeByte()`, `writeBytes()`, `writeDouble()`, etc. Purgez ensuite les données dans le tampon de sortie par le biais de la méthode `flush()`. Sur le serveur Telnet, les données sont écrites sur la connexion Socket à l'aide de la méthode `writeBytes()`, qui prend comme paramètre le tableau d'octets et l'envoie au tampon de sortie. La méthode `writeBytesToSocket()` se présente comme suit :

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

Cette méthode est appelée par la méthode `sendCommand()` du fichier d'application principal.

Affichage des messages provenant du serveur socket

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dès qu'un message est reçu du serveur socket ou qu'un événement se produit, la méthode personnalisée `msg()` est appelée. Cette méthode ajoute une chaîne au composant TextArea sur la scène et appelle une méthode `setScroll()` personnalisée, qui provoque le défilement vers le bas du composant TextArea. La méthode `msg()` se présente comme suit :

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

Si vous n'appliquez pas le défilement automatique au contenu du composant TextArea, les utilisateurs auront à le faire manuellement pour consulter la dernière réponse du serveur.

Défilement du contenu d'un composant TextArea

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `setScroll()` contient une seule ligne de code ActionScript qui permet de faire défiler verticalement le contenu du composant TextArea de manière que l'utilisateur puisse voir la dernière ligne de texte renvoyé. L'extrait de code suivant illustre la méthode `setScroll()` :

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

Cette méthode définit la propriété `verticalScrollPosition`, qui correspond au numéro de la ligne de caractère supérieure actuellement affichée, puis lui attribue la valeur de la propriété `maxVerticalScrollPosition`.

Sockets XML

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un socket XML permet de créer une connexion à un serveur distant qui demeure ouverte jusqu'à ce qu'elle soit explicitement fermée. Vous pouvez échanger des données au format chaîne (XML, par exemple) entre le serveur et le client. L'utilisation d'un serveur socket XML présente l'avantage de ne pas imposer au client de demander explicitement les données. Le serveur peut envoyer des données sans attendre de demande à chaque client connecté.

Dans Flash Player et les contenus Adobe AIR situés hors du sandbox d'application, les connexions socket XML nécessitent la présence d'un fichier de régulation socket sur le serveur cible. Pour plus d'informations, voir « Contrôles de site Web (fichiers de régulation) » à la page 1095 et « Connexion aux sockets » à la page 1114.

La classe XMLSocket ne peut pas automatiquement tunneler à travers les pare-feux car, contrairement au protocole RTMP (Real-Time Messaging Protocol), le XMLSocket n'a pas de capacité de tunneling HTTP. Si vous devez utiliser le tunneling HTTP, envisagez l'emploi de Flash Remoting ou Flash Media Server (qui prend en charge RTMP).

Les restrictions suivantes indiquent comment et où le contenu situé dans Flash Player ou une application AIR hors du sandbox de sécurité de l'application peut utiliser un objet XMLSocket pour se connecter au serveur :

- Pour le contenu hors du sandbox de sécurité de l'application, la méthode `XMLSocket.connect()` peut se connecter uniquement aux numéros de port TCP supérieurs ou égaux à 1024. En conséquence de cette restriction, les serveurs démon qui communiquent avec l'objet `XMLSocket` doivent également être affectés à des numéros de port supérieurs ou égaux à 1024. Les ports dont le numéro est inférieur à 1024 sont souvent utilisés par des services du système, tels que FTP (21), Telnet (23), SMTP (25), HTTP (80) et POP3 (110), de sorte que les objets XMLSocket ne puissent pas y accéder pour des raisons de sécurité. La restriction du numéro de port limite les possibilités d'accès à ces ressources et leur mauvaise utilisation.
- Pour le contenu hors du sandbox de sécurité de l'application, la méthode `XMLSocket.connect()` peut se connecter uniquement aux ordinateurs dans le même domaine de résidence du contenu. (Cette restriction est identique aux règles de sécurité de `URLLoader.load()`.) Pour se connecter à un serveur démon s'exécutant dans un domaine différent du domaine de résidence du contenu, vous pouvez créer sur le serveur un fichier de régulation interdomaines qui permette un accès à partir de domaines spécifiques. Pour plus d'informations sur les fichiers de régulation interdomaine, voir « Sécurité AIR » à la page 1122.

Remarque : *la configuration d'un serveur en vue de la communication avec un objet XMLSocket peut être difficile à réaliser. Si votre application ne requiert pas d'interactivité en temps réel, utilisez la classe URLLoader, plutôt que la classe XMLSocket.*

Vous pouvez utiliser les méthodes `XMLSocket.connect()` et `XMLSocket.send()` de la classe XMLSocket pour transférer un objet XML vers et à partir d'un serveur sur une connexion socket. La méthode `XMLSocket.connect()` établit une connexion socket avec le port d'un serveur Web. La méthode `XMLSocket.send()` transmet un objet XML au serveur spécifié dans la connexion socket.

Lorsque vous appelez la méthode `XMLSocket.connect()`, l'application ouvre une connexion TCP/IP avec le serveur et garde cette connexion ouverte jusqu'à ce que l'un des événements suivants se produise :

- La méthode `XMLSocket.close()` de la classe XMLSocket est appelée.
- Il n'existe plus aucune référence à l'objet XMLSocket.
- Flash Player se ferme.
- La connexion est interrompue (le modem est déconnecté, par exemple).

Connexion à un serveur par le biais de la classe XMLSocket

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour créer une connexion socket, vous devez créer une application côté serveur qui attend la demande de connexion socket et envoie une réponse à l'application Flash Player ou AIR. Ce type d'application côté serveur peut être programmé en AIR ou dans tout autre langage de programmation tel que Java, Python ou Perl. Pour utiliser la classe XMLSocket, l'ordinateur serveur doit exécuter un démon capable de lire le protocole simple utilisé par la classe XMLSocket :

- Les messages XML sont envoyés via une connexion socket à flux TCP/IP bidirectionnel simultané.
- Chaque message XML est un document XML complet, terminé par un octet nul (0).
- Un nombre illimité de messages XML peut être envoyé et reçu via une connexion XMLSocket.

Création d'un serveur socket XML Java et connexion à ce serveur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le code suivant illustre un simple serveur XMLSocket écrit en langage Java qui accepte les connexions entrantes et affichent les messages reçus dans la fenêtre d'invite de commande. Par défaut, un nouveau serveur est créé sur le port 8080 de votre machine locale, mais vous pouvez spécifier un numéro de port différent en lançant le serveur à partir de la ligne de commande.

Créez un document texte et ajoutez-y le code suivant :

```
import java.io.*;
import java.net.*;

class SimpleServer
{
    private static SimpleServer server;
    ServerSocket socket;
    Socket incoming;
    BufferedReader readerIn;
    PrintStream printOut;

    public static void main(String[] args)
    {
        int port = 8080;

        try
        {
            port = Integer.parseInt(args[0]);
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            // Catch exception and keep going.
        }

        server = new SimpleServer(port);
    }

    private SimpleServer(int port)
    {
        System.out.println(">> Starting SimpleServer");
        try
```

```
        {
            socket = new ServerSocket(port);
            incoming = socket.accept();
            readerIn = new BufferedReader(new InputStreamReader(incoming.getInputStream()));
            printOut = new PrintStream(incoming.getOutputStream());
            printOut.println("Enter EXIT to exit.\r");
            out("Enter EXIT to exit.\r");
            boolean done = false;
            while (!done)
            {
                String str = readerIn.readLine();
                if (str == null)
                {
                    done = true;
                }
                else
                {
                    out("Echo: " + str + "\r");
                    if(str.trim().equals("EXIT"))
                    {
                        done = true;
                    }
                }
                incoming.close();
            }
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }

    private void out(String str)
    {
        printOut.println(str);
        System.out.println(str);
    }
}
```

Enregistrez le document sur le disque dur sous le nom SimpleServer.java et compilez-le à l'aide d'un compilateur Java, qui crée un fichier de classe Java nommé SimpleServer.class.

Vous pouvez lancer le serveur XMLSocket via la ligne de commande en tapant `java SimpleServer`. Le fichier SimpleServer.class peut se situer à tout emplacement sur l'ordinateur local ou le réseau ; il n'est pas nécessaire de le placer dans le répertoire racine du serveur Web.

si vous ne pouvez pas lancer le serveur parce que des fichiers ne se trouvent pas dans le chemin de classe Java, essayez de le faire avec `java -classpath. SimpleServer`.

Pour établir une connexion au serveur XMLSocket à partir de l'application, vous devez créer une nouvelle occurrence de la classe XMLSocket et appeler la méthode `XMLSocket.connect()` tout en transférant le nom d'hôte et le numéro de port, comme suit :

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

Dès que vous recevez des données du serveur, l'événement `data` (`flash.events.DataEvent.DATA`) est distribué :

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

Pour envoyer des données au serveur XMLSocket, utilisez la méthode `XMLSocket.send()` et transmettez un objet XML ou une chaîne. Flash Player convertit le paramètre fourni en objet String et envoie au serveur XMLSocket son contenu suivi d'un octet nul (0) :

```
xmlsock.send(xmlFormattedData);
```

La méthode `XMLSocket.send()` ne renvoie pas de valeur indiquant si les données ont bien été transmises. Si une erreur se produit pendant la tentative d'envoi des données, une erreur IOError est renvoyée.

chaque message que vous envoyez au serveur socket doit se terminer par un caractère de changement de ligne (`\n`).

Pour plus d'informations, voir XMLSocket.

Sockets de serveur

Adobe AIR 2 et ultérieur

La classe ServerSocket permet d'autoriser d'autres processus à se connecter à l'application par le biais d'un socket TCP (Transport Control Protocol). Le processus de connexion peut s'exécuter sur l'ordinateur local ou un autre ordinateur connecté au réseau. Lorsqu'un objet ServerSocket reçoit une demande de connexion, il distribue un événement `connect`. L'objet ServerSocketConnectEvent distribué avec l'événement contient un objet Socket. Cet objet permet de communiquer ultérieurement avec l'autre processus.

Pour écouter les connexions socket entrantes :

1. Créez un objet ServerSocket et liez-le à un port local.
2. Ajoutez des écouteurs d'événements associés à l'événement `connect`.
3. Appelez la méthode `listen()`.
4. Répondez à l'événement `connect`, qui fournit un objet Socket pour chaque connexion entrante.

L'objet ServerSocket continue à écouter les nouvelles connexions jusqu'à ce que vous appeliez la méthode `close()`.

L'exemple de code suivant illustre la création d'une application de serveur socket. L'exemple écoute les connexions entrantes sur le port 8087. Lorsqu'une connexion est reçue, l'exemple envoie un message (la chaîne « Connected ») au socket client. Le serveur renvoie désormais tout message reçu au client.

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.IOErrorEvent;
    import flash.events.ProgressEvent;
    import flash.events.ServerSocketConnectEvent;
    import flash.net.ServerSocket;
    import flash.net.Socket;

    public class ServerSocketExample extends Sprite
    {
        private var serverSocket:ServerSocket;
        private var clientSockets:Array = new Array();

        public function ServerSocketExample()
        {
            try
            {
                // Create the server socket
                serverSocket = new ServerSocket();

                // Add the event listener
                serverSocket.addEventListener( Event.CONNECT, connectHandler );
                serverSocket.addEventListener( Event.CLOSE, onClose );

                // Bind to local port 8087
                serverSocket.bind( 8087, "127.0.0.1" );

                // Listen for connections
                serverSocket.listen();
                trace( "Listening on " + serverSocket.localPort );

            }
            catch(e:SecurityError)
            {
                trace(e);
            }
        }

        public function connectHandler(event:ServerSocketConnectEvent):void
        {
            //The socket is provided by the event object
            var socket:Socket = event.socket as Socket;
            clientSockets.push( socket );

            socket.addEventListener( ProgressEvent.SOCKET_DATA, socketDataHandler);
            socket.addEventListener( Event.CLOSE, onClientClose );
            socket.addEventListener( IOErrorEvent.IO_ERROR, onIOError );

            //Send a connect message
            socket.writeUTFBytes("Connected.");
            socket.flush();

            trace( "Sending connect message" );
        }
```

```
        public function socketDataHandler(event:ProgressEvent):void
        {
            var socket:Socket = event.target as Socket

            //Read the message from the socket
            var message:String = socket.readUTFBytes( socket.bytesAvailable );
            trace( "Received: " + message);

            // Echo the received message back to the sender
            message = "Echo -- " + message;
            socket.writeUTFBytes( message );
            socket.flush();
            trace( "Sending: " + message );
        }

        private function onClientClose( event:Event ):void
        {
            trace( "Connection to client closed." );
            //Should also remove from clientSockets array...
        }

        private function onIOError( errorEvent:IOErrorEvent ):void
        {
            trace( "IOError: " + errorEvent.text );
        }

        private function onClose( event:Event ):void
        {
            trace( "Server socket closed by OS." );
        }
}}
```

Pour plus d'informations, voir :

- ServerSocket
- ServerSocketConnectEvent
- Socket

Sockets UDP (AIR)

Adobe AIR 2 et ultérieur

Le protocole UDP (Universal Datagram Protocol) permet d'échanger des messages via une connexion réseau sans état. Non seulement UDP ne garantit pas la livraison dans l'ordre des messages, mais il ne peut pas se porter garant qu'ils soient livrés. Si UDP est activé, le code réseau du système d'exploitation consacre généralement moins de temps à la mise en ordre, au suivi et à la confirmation des messages. Par conséquent, les messages UDP sont le plus souvent livrés à l'application de destination plus rapidement que les messages TCP.

Les communications de type sockets UDP s'avèrent utiles lorsque vous devez envoyer des informations en temps réel telles qu'une actualisation des positions dans le cadre d'un jeu ou des paquets son dans une application de conversation vocale. Dans les applications de ce type, un certain pourcentage de perte de données est acceptable et une faible latence de transmission prime sur la livraison garantie. Dans quasiment tous les autres cas de figure, il est préférable d'utiliser les sockets TCP.

L'application AIR peut envoyer et recevoir des messages UDP à l'aide des classes DatagramSocket et DatagramSocketDataEvent. Pour envoyer ou recevoir un message UDP :

1. Créez un objet DatagramSocket.
2. Ajoutez un écouteur d'événements associé à l'événement `data`.
3. Liez le socket à une adresse IP locale et à un port par le biais de la méthode `bind()`.
4. Envoyez des messages en appelant la méthode `send()` et en transmettant l'adresse IP et le port de l'ordinateur cible.
5. Répondez à l'événement `data` pour recevoir des messages. L'objet DatagramSocketDataEvent distribué pour cet événement contient un objet ByteArray dans lequel figurent les données du message.

L'exemple de code suivant illustre l'envoi et la réception de messages UDP par une application. L'exemple envoie un message simple contenant la chaîne « Hello. » à l'ordinateur cible. Il effectue également le suivi du contenu de tout message reçu.

```
package
{
import flash.display.Sprite;
import flash.events.DatagramSocketDataEvent;
import flash.events.Event;
import flash.net.DatagramSocket;
import flash.utils.ByteArray;

public class DatagramSocketExample extends Sprite
{
    private var datagramSocket:DatagramSocket;

    //The IP and port for this computer
    private var localIP:String = "192.168.0.1";
    private var localPort:int = 55555;

    //The IP and port for the target computer
    private var targetIP:String = "192.168.0.2";
    private var targetPort:int = 55555;

    public function DatagramSocketExample()
    {
        //Create the socket
        datagramSocket = new DatagramSocket();
        datagramSocket.addEventListener( DatagramSocketDataEvent.DATA, dataReceived );

        //Bind the socket to the local network interface and port
```

```
        datagramSocket.bind( localPort, localIP );

        //Listen for incoming datagrams
        datagramSocket.receive();

        //Create a message in a ByteArray
        var data:ByteArray = new ByteArray();
        data.writeUTFBytes("Hello.");

        //Send the datagram message
        datagramSocket.send( data, 0, 0, targetIP, targetPort);
    }

    private function dataReceived( event:DatagramSocketDataEvent ):void
    {
        //Read the data from the datagram
        trace("Received from " + event.srcAddress + ":" + event.srcPort + "> " +
            event.data.readUTFBytes( event.data.bytesAvailable ) );
    }
}}
```

Tenez compte des considérations suivantes si vous utilisez des sockets UDP :

- Un paquet unique de données ne doit pas excéder l'unité maximale de transmission gérée par l'interface réseau ou tout nœud réseau entre l'expéditeur et le destinataire. L'intégralité des données de l'objet ByteArray transmise à la méthode send() est envoyée sous forme de paquet unique. (Dans TCP, les messages volumineux sont divisés en plusieurs paquets distincts.)
- Il ne se produit pas de négociation entre l'expéditeur et la cible. Les messages sont ignorés sans générer d'erreur si la cible n'existe pas ou qu'aucun écouteur actif n'est défini sur le port indiqué.
- Si vous faites appel à la méthode `connect()`, les messages issus d'autres sources sont ignorés. Une connexion UDP assure un filtrage des paquets uniquement. Cela ne signifie pas nécessairement qu'un processus d'écoute valide soit déployé à l'adresse et sur le port cibles.
- Le trafic UDP risque de submerger un réseau. En cas d'encombrement réseau, les administrateurs réseau sont parfois appelés à mettre en œuvre des contrôles de qualité de service. (TCP intègre des fonctions de régulation de la circulation en vue de réduire l'impact de l'encombrement réseau.)

Pour plus d'informations, voir :

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

Adresses IPv6

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player 9.0.115.0 (et versions ultérieures) prennent en charge IPv6 (Internet Protocol version 6). IPv6 est une version du protocole IP (Internet Protocol) qui prend en charge les adresses 128 bits (amélioration du protocole IPv4 précédent qui prend en charge les adresses 32 bits). Vous devrez peut-être activer IPv6 sur vos interfaces de mise en réseau. Pour plus d'informations, voir l'Aide du système d'exploitation hébergeant les données.

Si IPv6 est pris en charge sur le système hébergeant, vous pouvez spécifier des adresses littérales IPv6 numériques dans les URL entre crochets ([]), comme suit :

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

Flash Player renvoie les valeurs IPv6 littérales selon les règles suivantes :

- Flash Player renvoie la forme complète de la chaîne pour les adresses IPv6.
- La valeur IP ne possède pas d'abréviations à deux-points redoublés (::).
- Les chiffres hexadécimaux sont en bas de casse seulement.
- Les adresses IPv6 sont entourées de crochets ([]).
- Chaque quatuor d'adresses est représenté par 0 à 4 chiffres hexadécimaux, sans zéros non significatifs.
- Un quatuor d'adresses de zéros est représenté par un seul zéro (et pas un caractère de deux-points redoublé), sauf dans les cas suivants.

Les valeurs IPv6 que renvoie Flash Player sont soumises aux exceptions suivantes :

- Une adresse IPv6 non spécifiée (rien que des zéros) est représentée par [::].
- L'adresse de bouclage (loopback) ou localhost IPv6 est représentée par [::1].
- Les adresses mappées en IPv4 (converties en IPv6) sont représentées par [::ffff:a.b.c.d], où a.b.c.d constitue une valeur décimale à points (dotted-decimal) IPv4 classique.
- Les adresses compatibles avec IPv4 sont représentées par [::a.b.c.d], où a.b.c.d est une valeur décimale à points (dotted-decimal) IPv4 classique.

Chapitre 44 : Communications HTTP

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les applications Adobe® AIR® et Adobe® Flash® Player peuvent communiquer avec les serveurs HTTP pour charger des données, des images, des vidéos et échanger des messages.

Voir aussi

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

Chargement de données externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

ActionScript 3.0 comprend des mécanismes permettant de charger des données depuis des sources externes. Ces sources peuvent fournir un contenu statique, tel que des fichiers texte, ou un contenu dynamique généré par un script Web. Les données peuvent être formatées de plusieurs façons, et ActionScript propose une fonctionnalité permettant de décoder les données et d'y accéder. Vous pouvez également envoyer des données au serveur externe lors de leur récupération.

Utilisation de la classe URLRequest

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un grand nombre d'API qui chargent des données externes font appel à la classe URLRequest pour définir les propriétés de la requête réseau requise.

Propriétés de la classe URLRequest

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez définir les propriétés suivantes d'un objet URLRequest dans tout sandbox de sécurité :

Propriété	Description
contentType	Type de contenu MIME des données envoyées avec la requête d'URL. Si contentType n'est pas défini, les valeurs sont envoyées sous la forme `application/x-www-form-urlencoded`.
data	Objet contenant des données à transmettre avec la demande d'URL.
digest	Chaîne qui identifie de manière unique le composant de la plate-forme Adobe signé à enregistrer dans (ou extraire de) la mémoire cache d'Adobe® Flash® Player.
method	Méthode de requête HTTP, telle que GET ou POST. (Le contenu s'exécutant dans le domaine de sécurité de l'application AIR peut spécifier des chaînes autres que `"GET"` ou `"POST"` comme propriété `method`. Tous les verbes HTTP sont autorisés et `"GET"` est la méthode par défaut. Voir « Sécurité AIR » à la page 1122.)
requestHeaders	Tableau d'en-tête de requête HTTP à ajouter à la fin de la requête HTTP. Notez que l'autorisation de définir certains en-têtes est restreinte dans Flash Player, ainsi que dans un contenu AIR qui s'exécute en dehors du sandbox de sécurité d'application.
url	Spécifie l'URL qui fait l'objet de la requête.

AIR permet de définir d'autres propriétés de la classe URLRequest, dont ne dispose que le contenu AIR qui s'exécute dans le sandbox de sécurité d'application. Le contenu du sandbox d'application peut également définir des URL à l'aide des nouveaux modèles d'URL (parallèlement aux modèles standard tels que `file` et `http`).

Propriété	Description
followRedirects	Indique si des redirections sont utilisées (`true`, valeur par défaut) ou non (`false`). La définition de cette propriété n'est gérée que dans le sandbox d'une application AIR.
manageCookies	Indique si la pile du protocole HTTP doit gérer les cookies (`true`, valeur par défaut) ou non (`false`) pour cette requête. La définition de cette propriété n'est gérée que dans le sandbox d'une application AIR.
authenticate	Indique si les requêtes d'authentification doivent être traitées (`true`) pour cette requête. La définition de cette propriété n'est gérée que dans le sandbox d'une application AIR. Par défaut, les requêtes sont authentifiées ; il est par conséquent possible qu'une boîte de dialogue d'authentification s'affiche si le serveur requiert des informations d'identification. Vous pouvez également définir le nom d'utilisateur et le mot de passe par le biais de la classe URLRequestDefaults (voir « Définition des paramètres par défaut des objets URLRequest (AIR uniquement) » à la page 844).
cacheResponse	Indique si les données de réponse doivent être mises en mémoire cache pour cette requête. La définition de cette propriété n'est gérée que dans le sandbox d'une application AIR. Par défaut, la réponse est mise en mémoire cache (`true`).
useCache	Indique si le cache local doit être consulté avant que la classe URLRequest récupère les données. La définition de cette propriété n'est gérée que dans le sandbox d'une application AIR. Par défaut, le cache local doit être utilisé, s'il est disponible (`true`).
userAgent	Indique la chaîne agent utilisateur à utiliser dans la requête HTTP.

Remarque : *la classe HTMLLoader possède des propriétés associées pour les paramètres associés au contenu chargé par un objet HTMLLoader. Pour plus d'informations, voir « A propos de la classe HTMLLoader » à la page 1019.*

Définition des paramètres par défaut des objets URLRequest (AIR uniquement)

Adobe AIR 1.0 et les versions ultérieures

La classe URLRequestDefaults permet de définir les paramètres par défaut propres à une application des objets URLRequest. Par exemple, le code suivant définit les valeurs par défaut des propriétés `manageCookies` et `useCache`. Tous les nouveaux objets URLRequest utiliseront les valeurs spécifiées de ces propriétés au lieu des valeurs par défaut standard :

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

Remarque : *la classe URLRequestDefaults est réservée au contenu qui s'exécute dans Adobe AIR. Elle n'est pas prise en charge dans Flash Player.*

La classe URLRequestDefaults inclut une méthode `setLoginCredentialsForHost()` qui vous permet de spécifier un nom d'utilisateur et un mot de passe par défaut pour un hôte spécifique. L'hôte, défini dans le paramètre hostname de la méthode, peut être un domaine, tel que `"www.example.com"`, ou un domaine et un numéro de port, par exemple `"www.example.com:80"`. Notez que `"example.com"`, `"www.example.com"` et `"sales.example.com"` sont considérés comme hôtes uniques.

Ces informations d'identification ne sont utilisées que lorsque le serveur les demande. Si l'utilisateur s'est déjà authentifié (à l'aide de la boîte de dialogue d'authentification, par exemple), appeler la méthode `setLoginCredentialsForHost()` ne modifie pas l'utilisateur authentifié.

Le code suivant permet de définir le nom d'utilisateur et le mot de passe par défaut à utiliser pour les requêtes envoyées à www.example.com :

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

Les paramètres URLRequestDefaults ne s'appliquent qu'au domaine d'application actuel, à une exception près. Les informations d'identification transmises à la méthode `setLoginCredentialsForHost()` sont utilisées pour les requêtes effectuées dans tout domaine de l'application AIR.

Pour plus d'informations, voir la classe URLRequestDefaults dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Modèles d'URI

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les modèles d'URI standard, tel le modèle ci-dessous, peuvent être utilisés dans les requêtes effectuées à partir de tout sandbox de sécurité :

http: et https:
A utiliser pour les URL Internet standard (comme elles sont utilisées dans un navigateur Web).

file:
Utilisez `file:` pour spécifier l'URL d'un fichier qui réside dans le système de fichiers local. Exemple :

```
file:///c:/AIR Test/test.txt
```

Dans AIR, vous disposez également des modèles suivants lorsque vous définissez l'URL d'un contenu s'exécutant dans le sandbox de sécurité de l'application :

app:
Le modèle `app:` permet de spécifier un chemin relatif au répertoire racine de l'application installée. Par exemple, le chemin suivant pointe vers un sous-répertoire de ressources du répertoire de l'application installée :

```
app:/resources
```

Lorsqu'une application AIR est lancée à l'aide de l'application de débogage du lanceur AIR (ADL), le répertoire d'application correspond à celui qui contient le fichier descripteur d'application.

L'URL (et la propriété `url`) d'un objet File créé avec `File.applicationDirectory` fait appel au modèle d'URI `app`, comme dans l'exemple suivant :

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:

Le modèle `app:storage` permet de spécifier un chemin relatif au répertoire de stockage de données de l'application. Pour chaque application installée (et utilisateur), AIR crée un répertoire de stockage d'application unique. Cet emplacement est utile pour stocker des données spécifiques à l'application concernée. Par exemple, le chemin suivant pointe vers le fichier prefs.xml, qui réside dans le sous-répertoire settings du répertoire de stockage de l'application :

```
app-storage:/settings/prefs.xml
```

L'URL (et la propriété `url`) d'un objet File créé avec `File.applicationStorageDirectory` utilise le modèle d'URI `app-storage`, comme suit :

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto :

Vous pouvez utiliser le modèle mailto dans les objets URLRequest transmis à la fonction `navigateToURL()`. Voir « Ouverture d'une URL dans une autre application » à la page 860.

Vous pouvez faire appel à un objet URLRequest basé sur l'un de ces modèles d'URI pour définir la requête d'URL d'un certain nombre d'objets, tels qu'un objet FileStream ou Sound. Vous pouvez par ailleurs utiliser ces modèles dans le contenu HTML qui s'exécute dans AIR. Il est ainsi possible d'y faire appel dans l'attribut `src` d'une balise `img`.

Néanmoins, vous ne pouvez utiliser ces modèles d'URI spécifiques à AIR (`app:` et `app-storage:`) que dans le contenu du sandbox de sécurité de l'application. Pour plus d'informations, voir « Sécurité AIR » à la page 1122.

Définition des variables URL

Bien qu'il soit possible d'ajouter directement des variables à la chaîne de l'URL, il s'avère souvent plus facile de définir toute variable requise par une requête à l'aide de la classe URLVariables.

Les méthodes permettant d'ajouter des paramètres à un objet URLVariables sont au nombre de trois :

- Au sein du constructeur URLVariables
- A l'aide de la méthode `URLVariables.decode()`
- Sous forme de propriétés dynamiques de l'objet URLVariables en tant que tel

L'exemple suivant illustre les trois méthodes et indique comment affecter les variables à un objet URLRequest :

```
var urlVar:URLVariables = new URLVariables( "one=1&two=2" );
urlVar.decode("amp=" + encodeURIComponent( "&" ) );
urlVar.three = 3;
urlVar.amp2 = "&&";
trace(urlVar.toString()); //amp=%26&amp2=%26%26&one=1&two=2&three=3

var urlRequest:URLRequest = new URLRequest( "http://www.example.com/test.cfm" );
urlRequest.data = urlVar;
```

Lorsque vous définissez des variables au sein du constructeur URLVariables ou à l'aide de la méthode `URLVariables.decode()`, veillez à coder au format URL les caractères dont la signification est particulière dans une chaîne d'URI. Ainsi, lorsque vous insérez une esperluette dans un nom de paramètre ou une valeur, vous devez la coder de sorte à remplacer `&` par `%26`, car l'esperluette fait office de délimiteur de paramètres. Vous disposez à cet effet de la fonction de niveau supérieur `encodeURIComponent()`.

Utilisation de la classe URLLoader

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe URLLoader permet d'envoyer une requête à un serveur et d'accéder aux informations renvoyées. Vous disposez également de la classe URLLoader pour accéder aux fichiers du système de fichiers local dans les contextes où l'accès aux fichiers locaux est autorisé (le sandbox local avec système de fichiers de Flash Player et le sandbox d'application AIR, par exemple). La classe URLLoader télécharge des données à partir d'une URL sous forme de texte, de données binaires ou de variables codées au format URL. La classe URLLoader distribue des événements tels que `complete`, `httpStatus`, `ioError`, `open`, `progress` et `securityError`.

Le modèle de gestion des événements d'ActionScript 3.0 est sensiblement différent du modèle d'ActionScript 2.0, qui utilisait les gestionnaires d'événement `LoadVars.onData`, `LoadVars.onHTTPStatus` et `LoadVars.onLoad`. Pour plus d'informations sur la gestion des événements en ActionScript 3.0, voir « Gestion des événements » à la page 129.

Les données téléchargées ne sont pas disponibles tant que le téléchargement n'est pas terminé. Pour surveiller la progression du téléchargement (nombre d'octets chargés et nombre total d'octets), écoutez la distribution de l'événement `progress`. Toutefois, si le chargement d'un fichier est rapide, l'événement `progress` risque de ne pas être distribué. Une fois que le téléchargement d'un fichier est terminé, l'événement `complete` est distribué. En définissant la propriété `dataFormat` de la classe URLLoader, vous pouvez recevoir les données sous forme de texte, de données binaires brutes ou d'objet URLVariables.

La méthode `URLLoader.load()` (et éventuellement le constructeur de la classe URLLoader) gère un seul paramètre, `request`, qui correspond à un objet URLRequest. Un objet URLRequest contient toutes les informations d'une requête HTTP unique, telles que l'URL cible, la méthode de requête (`GET` ou `POST`), d'autres informations d'en-tête et le type MIME.

Pour charger un paquet XML dans un script côté serveur, par exemple, vous pouvez utiliser le code ci-après :

```
var secondsUTC:Number = new Date().time;
var dataXML:XML =
    <clock>
        <time>{secondsUTC}</time>
    </clock>;
var request:URLRequest = new URLRequest("http://www.yourdomain.com/time.cfm");
request.contentType = "text/xml";
request.data = dataXML.toXMLString();
request.method = URLRequestMethod.POST;
var loader:URLLoader = new URLLoader();
loader.load(request);
```

L'extrait ci-dessus crée un document XML appelé `dataXML`, qui contient le paquet XML à envoyer au serveur. L'exemple définit la propriété `contentType` de l'objet URLRequest sur `"text/xml"` et affecte le document XML à la propriété `data` de l'objet URLRequest. Il crée enfin un objet URLLoader et envoie la requête au script distant par le biais de la méthode `load()`.

Utilisation de la classe URLStream

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe URLStream permet d'accéder aux données en cours de téléchargement au fur et à mesure de leur arrivée. Elle permet également de fermer un flux continu avant la fin du téléchargement. Les données téléchargées sont disponibles au format binaire brut.

Lorsque vous lisez les données issues d'un objet URLStream, faites appel à la propriété `bytesAvailable` pour déterminer si les données disponibles sont suffisantes avant de les lire. Une exception EOFError est renvoyée si vous essayez de lire des données qui ne sont pas disponibles.

Evénement httpResponseStatus (AIR)
Dans Adobe AIR, la classe URLStream distribue un événement `httpResponseStatus` en plus de l'événement `httpStatus`. L'événement `httpResponseStatus` est envoyé avant toute donnée de réponse. L'événement `httpResponseStatus` (représenté par la classe HTTPStatusEvent) inclut une propriété `responseURL`, qui correspond à l'URL que la réponse a renvoyée, et une propriété `responseHeaders`, qui est un tableau d'objets URLRequestHeader représentant les en-têtes de réponse que la réponse a renvoyés.

Chargement de données à partir de documents externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous programmez des applications dynamiques, il peut s'avérer utile de charger des données issues de fichiers externes ou de scripts côté serveur. Vous pouvez ainsi programmer des applications dynamiques sans avoir à les modifier ou les recompiler. Par exemple, si vous créez une application « conseil du jour », vous pouvez écrire un script côté serveur qui récupère un conseil au hasard dans une base de données et l'enregistre dans un fichier texte une fois par jour. Votre application peut ensuite charger le contenu du fichier texte statique au lieu d'envoyer une requête à la base de données à chaque fois.

L'extrait de code suivant crée un objet URLRequest et URLLoader, qui charge le contenu d'un fichier texte externe nommé params.txt :

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

Par défaut, si vous ne définissez aucune méthode de requête, Flash Player et AIR chargent le contenu à l'aide de la méthode HTTP `GET`. Pour envoyer la requête avec la méthode `POST`, définissez la propriété `request.method` sur `POST` à l'aide de la constante statique `URLRequestMethod.POST`, comme l'illustre le code suivant :

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

Le document externe, params.txt, chargé au moment de l'exécution, contient les données suivantes :

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,Dece
mber&dayNames=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Le fichier contient deux paramètres, `monthNames` et `dayNames`. Chacun des paramètres contient une liste séparée par des virgules qui est analysée sous forme de chaîne. Vous pouvez transformer cette liste en tableau à l'aide de la méthode `String.split()`.

Evitez d'utiliser des mots réservés ou des éléments de langage comme noms de variables dans les fichiers de données externes, car cela complique la lecture et le débogage du code.

Une fois les données chargées, l'événement `complete` est distribué et le contenu du document externe peut alors être utilisé dans la propriété `data` de URLLoader, comme l'illustre le code suivant :

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

Si le document distant contient des paires nom-valeur, vous pouvez analyser les données à l'aide de la classe URLVariables en transmettant le contenu du fichier chargé, comme suit :

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

Chaque paire nom-valeur issue du fichier externe devient une nouvelle propriété de l'objet URLVariables. Chaque propriété de cet objet dans l'exemple de code précédent est traité comme une chaîne. Si la valeur de la paire nom-valeur est une liste d'éléments, vous pouvez convertir la chaîne en tableau en appelant la méthode `String.split()`, comme suit :

```
var dayNameArray:Array = variables.dayNames.split(",");
```

Si vous chargez des données numériques à partir de fichiers externes, convertissez-les en valeurs numériques à l'aide d'une fonction de niveau supérieur, telle que `int()`, `uint()` ou `Number()`.

Au lieu de charger le contenu du fichier distant sous forme de chaîne et de créer un objet URLVariables, vous pouvez attribuer à la propriété `URLLoader.dataFormat` la valeur de l'une des propriétés statiques de la classe URLLoaderDataFormat. Les trois valeurs possibles de la propriété `URLLoader.dataFormat` sont les suivantes :

- `URLLoaderDataFormat.BINARY` : la propriété `URLLoader.data` contient des données binaires stockées dans un objet ByteArray.
- `URLLoaderDataFormat.TEXT` : la propriété `URLLoader.data` contient du texte sous forme d'objet String.
- `URLLoaderDataFormat.VARIABLES` : la propriété `URLLoader.data` contient des variables encodées au format URL issues d'un objet URLVariables.

Le code suivant montre comment vous pouvez automatiquement analyser les données chargées dans un objet URLVariables en attribuant à la propriété `URLLoader.dataFormat` la valeur `URLLoaderDataFormat.VARIABLES`.

```
package
{
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLLoader;
    import flash.net.URLLoaderDataFormat;
    import flash.net.URLRequest;

    public class URLLoaderDataFormatExample extends Sprite
    {
        public function URLLoaderDataFormatExample()
        {
            var request:URLRequest = new URLRequest("http://www.[yourdomain].com/params.txt");
            var variables:URLLoader = new URLLoader();
            variables.dataFormat = URLLoaderDataFormat.VARIABLES;
            variables.addEventListener(Event.COMPLETE, completeHandler);
            try
            {
                variables.load(request);
            }
            catch (error:Error)
            {
                trace("Unable to load URL: " + error);
            }
        }
        private function completeHandler(event:Event):void
        {
        var loader:URLLoader = URLLoader(event.target);
        trace(loader.data.dayNames);
        }
    }
}
```

Remarque : *la valeur par défaut de* `URLLoader.dataFormat` *est* `URLLoaderDataFormat.TEXT`.

Comme le montre l'exemple suivant, le chargement de données XML à partir d'un fichier externe s'effectue comme le chargement de données URLVariables. Vous pouvez créer une occurrence d'URLRequest et une occurrence d'URLLoader et vous en servir pour télécharger un document XML distant. Lorsque le fichier est complètement téléchargé, l'événement `Event.COMPLETE` est distribué et le contenu du fichier externe est converti en occurrence de XML, que vous pouvez analyser avec les méthodes et propriétés XML.

```
package
{
    import flash.display.Sprite;
    import flash.errors.*;
    import flash.events.*;
    import flash.net.URLLoader;
    import flash.net.URLRequest;

    public class ExternalDocs extends Sprite
    {
        public function ExternalDocs()
        {
            var request:URLRequest = new URLRequest("http://www.[yourdomain].com/data.xml");
            var loader:URLLoader = new URLLoader();
            loader.addEventListener(Event.COMPLETE, completeHandler);
            try
            {
                loader.load(request);
            }
            catch (error:ArgumentError)
            {
                trace("An ArgumentError has occurred.");
            }
            catch (error:SecurityError)
            {
                trace("A SecurityError has occurred.");
            }
        }
        private function completeHandler(event:Event):void
        {
            var dataXML:XML = XML(event.target.data);
            trace(dataXML.toXMLString());
        }
    }
}
```

Communication avec des scripts externes

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Outre le chargement de fichiers de données externes, la classe URLVariables permet d'envoyer des variables à un script côté serveur et de traiter la réponse du serveur. Cela s'avère utile, par exemple, si vous programmez un jeu et souhaitez envoyer le score de l'utilisateur à un serveur afin de vérifier s'il faut l'ajouter à la liste des scores les plus élevés ; ou encore envoyer les informations de connexion de l'utilisateur au serveur pour validation. Un script côté serveur peut traiter le nom d'utilisateur et le mot de passe, les comparer à une base de données et confirmer en retour la validité des informations fournies par l'utilisateur.

L'extrait de code ci-après crée un objet URLVariables nommé `variables`, qui génère une nouvelle variable appelée `name`. Ensuite, un objet URLRequest est créé pour spécifier l'URL du script côté serveur à laquelle doivent être envoyées les variables. Vous pouvez alors définir la propriété `method` de l'objet URLRequest pour envoyer les variables sous forme de requête HTTP `POST` . Pour ajouter l'objet URLVariables à la requête URL, définissez la propriété `data` de l'objet URLRequest sur l'objet URLVariables créé plus tôt. Enfin, l'occurrence de URLLoader est créée et la méthode `URLLoader.load()` appelée ; cette dernière lance la requête.

```
var variables:URLVariables = new URLVariables("name=Franklin");
var request:URLRequest = new URLRequest();
request.url = "http://www.[yourdomain].com/greeting.cfm";
request.method = URLRequestMethod.POST;
request.data = variables;
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.VARIABLES;
loader.addEventListener(Event.COMPLETE, completeHandler);
try
{
    loader.load(request);
}
catch (error:Error)
{
    trace("Unable to load URL");
}

function completeHandler(event:Event):void
{
    trace(event.target.data.welcomeMessage);
}
```

Le code suivant reprend le contenu du document Adobe ColdFusion® greeting.cfm utilisé dans l'exemple précédent :

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

Requêtes de service Web

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous disposez d'un large éventail de services Web HTTP. Les types principaux sont les suivants :

- REST
- XML-RPC
- SOAP

Pour appeler un service Web en ActionScript 3, vous créez un objet URLRequest, vous créez l'appel au service Web par le biais de variables URL ou d'un document XML, puis vous envoyez l'appel au service à l'aide d'un objet URLLoader. La structure Flex contient plusieurs classes qui facilitent l'utilisation des services Web, ce qui s'avère particulièrement utile pour accéder à des services SOAP complexes. Depuis Flash Professional CS3, vous pouvez faire appel aux classes Flex dans les applications développées dans Flash Professional et dans Flash Builder.

Dans les applications AIR HTML, vous disposez des classes URLRequest et URLLoader ou de la classe XMLHttpRequest de JavaScript. Le cas échéant, vous pouvez également créer une bibliothèque SWF qui expose les composants du service Web de la structure Flex au code JavaScript.

Si l'application s'exécute dans un navigateur, vous ne pouvez utiliser les services Web que s'ils tournent dans le même domaine Internet que le fichier SWF appelant, sauf si le serveur qui héberge le service Web héberge également un fichier de régulation interdomaines qui autorise l'accès à partir d'autres domaines. Une technique souvent utilisée en l'absence d'un fichier de régulation interdomaines consiste à traiter par proxy les requêtes via votre propre serveur. Adobe Blaze DS et Adobe LiveCycle prennent en charge le traitement proxy des services Web.

Dans les applications AIR, un fichier de régulation interdomaines n'est pas obligatoire si l'appel au service Web provient du sandbox de sécurité d'application. Un contenu d'application AIR n'est jamais transmis par un serveur qui réside dans un domaine distant. Il ne peut donc pas participer aux types d'attaques bloquées par les régulations interdomaines. Dans les applications AIR HTML, un contenu situé dans le sandbox de sécurité d'application peut générer des requêtes XMLHttpRequests interdomaines. Vous pouvez autoriser un contenu situé dans d'autres sandbox de sécurité à générer des requêtes XMLHttpRequests interdomaines, sous réserve que le contenu soit chargé dans un iframe.

Voir aussi

« Contrôles de site Web (fichiers de régulation) » à la page 1095

Adobe BlazeDS

Adobe LiveCycle ES2

Architecture REST

XML-RPC

Protocole SOAP

Requêtes de service Web de type REST

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les services Web de type REST font appel aux verbes d'une méthode HTTP pour désigner l'action de base et aux variables URL pour spécifier les détails correspondants. Ainsi, une requête visant à obtenir les données associées à un élément peut utiliser le verbe GET et les variables URL pour spécifier un nom de méthode et un ID d'élément. La chaîne d'URL qui en résulte est similaire au code ci-dessous :

```
http://service.example.com/?method=getItem&id=d3452
```

Pour accéder à un service Web de type REST en ActionScript, vous disposez des classes URLRequest, URLVariables et URLLoader. Si le code JavaScript est intégré à une application AIR, vous pouvez également faire appel à une requête XMLHttpRequest.

La programmation d'un appel au service Web de type REST en ActionScript implique généralement la procédure suivante :

1. Créez un objet URLRequest.
2. Associez l'URL du service et le verbe de la méthode HTTP à l'objet de la requête.
3. Créez un objet URLVariables.
4. Définissez les paramètres de l'appel au service sous forme de propriétés dynamiques de l'objet variables.
5. Affectez l'objet variables à la propriété data de l'objet de la requête.
6. Envoyez l'appel au service à l'aide d'un objet URLLoader.

7 Gérez l'événement `complete` distribué par l'objet URLLoader qui indique que l'appel au service est terminé. Il est également recommandé d'écouter les divers événements d'erreur susceptibles d'être distribués par un objet URLLoader.

Considérons par exemple un service Web exposant une méthode test qui renvoie les paramètres de l'appel au demandeur. Le code ActionScript suivant pourrait permettre d'appeler le service :

```
import flash.events.Event;
import flash.events.ErrorEvent;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLLoader;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLVariables;

private var requestor:URLLoader = new URLLoader();
public function restServiceCall():void
{
    //Create the HTTP request object
    var request:URLRequest = new URLRequest( "http://service.example.com/" );
    request.method = URLRequestMethod.GET;

    //Add the URL variables
    var variables:URLVariables = new URLVariables();
    variables.method = "test.echo";
    variables.api_key = "123456ABC";
    variables.message = "Able was I, ere I saw Elba.";
    request.data = variables;

    //Initiate the transaction
    requestor = new URLLoader();
    requestor.addEventListener( Event.COMPLETE, httpRequestComplete );
    requestor.addEventListener( IOErrorEvent.IOERROR, httpRequestError );
    requestor.addEventListener( SecurityErrorEvent.SECURITY_ERROR, httpRequestError );
    requestor.load( request );
}
private function httpRequestComplete( event:Event ):void
{
    trace( event.target.data );
}

private function httpRequestError( error:ErrorEvent ):void{
    trace( "An error occured: " + error.message );
}
```

Si le code JavaScript est intégré à une application AIR, vous pouvez effectuer la même requête par le biais de l'objet XMLHttpRequest :

```
<html>
<head><title>RESTful web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    //Create a conveninece object to hold the call properties
    var request = {};
    request.URL = "http://service.example.com/";
    request.method = "test.echo";
    request.HTTPmethod = "GET";
    request.parameters = {};
    request.parameters.api_key = "ABCDEF123";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestURL = makeURL( request );
    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, requestURL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);

    requestDisplay.innerHTML = requestURL;
}
//Convert the request object into a properly formatted URL
function makeURL( request )
{
    var url = request.URL + "?method=" + escape( request.method );
    for( var property in request.parameters )
    {
        url += "&" + property + "=" + escape( request.parameters[property] );
    }

    return url;
}
</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<div id="request"></div>
<h1>Result:</h1>
<div id="result"></div>
</body>
</html>
```

Requêtes de service Web de type XML-RPC

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un service Web de type XML-RPC utilise en tant que paramètres d'appel un document XML au lieu d'un jeu de variables URL. Pour effectuer une transaction avec un service Web de type XML-RPC, créez un message XML correctement formaté et envoyez-le au service Web par le biais de la méthode HTTP `POST`. Définissez en outre l'en-tête `Content-Type` de la requête de sorte que le serveur traite les données de cette dernière comme des données XML.

L'exemple suivant illustre l'utilisation du même appel au service Web que l'exemple REST, mais il s'agit à présent d'un service XML-RPC :

```
import flash.events.Event;
import flash.events.ErrorEvent;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLLoader;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLVariables;
public function xmlRPCRequest():void
{
    //Create the XML-RPC document
    var xmlRPC:XML = <methodCall>
                                    <methodName></methodName>
                                    <params>
                                        <param>
                                            <value>
                                                <struct/>
                                            </value>
                                        </param>
                                    </params>
                                </methodCall>;

    xmlRPC.methodName = "test.echo";

    //Add the method parameters
    var parameters:Object = new Object();
    parameters.api_key = "123456ABC";
    parameters.message = "Able was I, ere I saw Elba.";

    for( var propertyName:String in parameters )
    {
        xmlRPC..struct.member[xmlRPC..struct.member.length + 1] =
            <member>
                <name>{propertyName}</name>
                <value>
                    <string>{parameters[propertyName]}</string>
                </value>
            </member>;
    }

    //Create the HTTP request object
    var request:URLRequest = new URLRequest( "http://service.example.com/xml-rpc/" );
    request.method = URLRequestMethod.POST;
    request.cacheResponse = false;
    request.requestHeaders.push(new URLRequestHeader("Content-Type", "application/xml"));
```

```
    request.data = xmlRPC;

    //Initiate the request
    requestor = new URLLoader();
    requestor.dataFormat = URLLoaderDataFormat.TEXT;
    requestor.addEventListener( Event.COMPLETE, xmlRPCRequestComplete );
    requestor.addEventListener( IOErrorEvent.IO_ERROR, xmlRPCRequestError );
    requestor.addEventListener( SecurityErrorEvent.SECURITY_ERROR, xmlRPCRequestError );
    requestor.load( request );
}

private function xmlRPCRequestComplete( event:Event ):void
{
    trace( XML(event.target.data).toXMLString() );
}

private function xmlRPCRequestError( error:ErrorEvent ):void
{
    trace( "An error occurred: " + error );
}
```

Dans AIR, WebKit ne prend pas en charge la syntaxe E4X. La méthode utilisée pour créer le document XML dans l'exemple précédent ne fonctionne par conséquent pas dans un code JavaScript. Vous devez utiliser les méthodes DOM pour créer le document XML ou créer celui-ci sous forme de chaîne et la convertir au format XML à l'aide de la classe DOMParser de JavaScript.

L'exemple suivant fait appel aux méthodes DOM pour créer un message XML-RPC et à une requête XMLHttpRequest pour effectuer la transaction avec le service Web :

```
<html>
<head>
<title>XML-RPC web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    var request = {};
    request.URL = "http://services.example.com/xmlrpc/";
    request.method = "test.echo";
    request.HTTPmethod = "POST";
    request.parameters = {};
    request.parameters.api_key = "123456ABC";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestMessage = formatXMLRPC( request );

    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, request.URL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerText = xmlhttp.responseText;
        }
    }
    xmlhttp.send( requestMessage );
```

```
    requestDisplay.innerText = xmlToString( requestMessage.documentElement );
}

//Formats a request as XML-RPC document
function formatXMLRPC( request )
{
    var xmldoc = document.implementation.createDocument( "", "", null );
    var root = xmldoc.createElement( "methodCall" );
    xmldoc.appendChild( root );
    var methodName = xmldoc.createElement( "methodName" );
    var methodString = xmldoc.createTextNode( request.method );
    methodName.appendChild( methodString );

    root.appendChild( methodName );

        var params = xmldoc.createElement( "params" );
        root.appendChild( params );

        var param = xmldoc.createElement( "param" );
        params.appendChild( param );
        var value = xmldoc.createElement( "value" );
        param.appendChild( value );
        var struct = xmldoc.createElement( "struct" );
        value.appendChild( struct );

        for( var property in request.parameters )
        {
            var member = xmldoc.createElement( "member" );
            struct.appendChild( member );

            var name = xmldoc.createElement( "name" );
            var paramName = xmldoc.createTextNode( property );
            name.appendChild( paramName )
            member.appendChild( name );

            var value = xmldoc.createElement( "value" );
            var type = xmldoc.createElement( "string" );
            value.appendChild( type );
            var paramValue = xmldoc.createTextNode( request.parameters[property] );
            type.appendChild( paramValue )
            member.appendChild( value );
        }
    return xmldoc;
}

//Returns a string representation of an XML node
function xmlToString( rootNode, indent )
{
    if( indent == null ) indent = "";
    var result = indent + "<" + rootNode.tagName + ">\n";
    for( var i = 0; i < rootNode.childNodes.length; i++)
    {
        if(rootNode.childNodes.item( i ).nodeType == Node.TEXT_NODE )
        {
            result += indent + "    " + rootNode.childNodes.item( i ).textContent + "\n";
        }
    }
```

```
    if( rootNode.childElementCount > 0 )
    {
        result += xmlToString( rootNode.firstElementChild, indent + "    " );
    }
    if( rootNode.nextElementSibling )
    {
        result += indent + "</" + rootNode.tagName + ">\n";
        result += xmlToString( rootNode.nextElementSibling, indent );
    }
    else
    {
        result += indent +"</" + rootNode.tagName + ">\n";
    }
    return result;
}

</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<pre id="request"></pre>
<h1>Result:</h1>
<pre id="result"></pre>
</body>
</html>
```

Requêtes de service Web de type SOAP

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Basé sur le concept de service Web XML-RPC général, SOAP propose des techniques plus évoluées, bien que plus complexes, pour transférer les données d'un type déterminé. Les services Web SOAP fournissent généralement un fichier WSDL (Web Service Description Language) qui spécifie les appels au service Web, les types de données et l'URL du service. Bien qu'ActionScript 3 ne prenne pas directement en charge SOAP, vous pouvez créer « manuellement » un message XML SOAP, le publier sur le serveur, puis analyser les résultats. Toutefois, à l'exception du service Web SOAP le plus simple, vous gagnerez probablement un temps considérable en phase de développement si vous utilisez une bibliothèque SOAP existante.

La structure Flex intègre des bibliothèques permettant d'accéder aux services Web SOAP. Dans Flash Builder, la bibliothèque, rpc.swc, est automatiquement intégrée aux projets Flex, puisqu'elle fait partie de la structure Flex. Dans Flash Professional, vous pouvez ajouter les fichiers Flex framework.swc et rpc.swc au chemin de la bibliothèque d'un projet, puis accéder aux classes Flex à l'aide d'ActionScript.

Voir aussi

Utilisation du composant de service Web Flex dans Flash Professional

Christophe Coenraets : Real-time Trader Desktop for Android (disponible en anglais uniquement)

Ouverture d'une URL dans une autre application

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous disposez de la fonction `navigateToURL()` pour ouvrir une URL dans un navigateur Web ou une autre application. Si le contenu s'exécute dans AIR, la fonction `navigateToURL()` ouvre la page dans le navigateur Web par défaut du système.

Pour l'objet URLRequest que vous transmettez comme paramètre `request` de cette fonction, seule la propriété `url` est utilisée.

Le premier paramètre de la fonction `navigateToURL()`, le paramètre `navigate`, est un objet URLRequest (voir « Utilisation de la classe URLRequest » à la page 843). Le deuxième paramètre est un paramètre `window` facultatif, dans lequel vous pouvez spécifier le nom de la fenêtre. Le code suivant ouvre par exemple la page Web www.adobe.com :

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

Remarque : *lorsque vous utilisez la fonction `navigateToURL()`, le moteur d'exécution traite un objet URLRequest qui utilise la méthode POST (celui dont la propriété `method` est définie sur `URLRequestMethod.POST`) comme la méthode GET.*

Si vous utilisez la fonction `navigateToURL()`, les modèles d'URI sont autorisés en fonction du sandbox de sécurité du code qui appelle la fonction `navigateToURL()`.

Certaines API permettent de lancer le contenu dans un navigateur Web. Pour des raisons de sécurité, certains modèles d'URI sont interdits lors de l'utilisation de ces API dans AIR. La liste des modèles non autorisés dépend du sandbox de sécurité du code utilisant l'API. (Pour plus d'informations sur les sandbox de sécurité, voir le chapitre « Sécurité AIR » à la page 1122.)

Sandbox de l'application (AIR uniquement)

Vous pouvez utiliser n'importe quel modèle d'URI dans l'URL activée par le contenu s'exécutant dans le sandbox de l'application AIR. Une application doit être enregistrée pour gérer le modèle d'URI. Dans le cas contraire, la demande n'aboutit pas. Les modèles suivants sont pris en charge sur de nombreux périphériques et ordinateurs :

- `http:`
- `https:`
- `file:`
- `mailto:` AIR dirige ces requêtes à l'application de messagerie système enregistrée
- `sms:` — AIR envoie les requêtes `sms:` à l'application de message de texte par défaut. Le format URL doit respecter les conventions du système sur lequel s'exécute l'application. Par exemple, sur Android, le modèle d'URI doit être en minuscules.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- `tel:` — AIR envoie les requêtes `tel:` à l'application de numérotation téléphonique par défaut. Le format URL doit respecter les conventions du système sur lequel s'exécute l'application. Par exemple, sur Android, le modèle d'URI doit être en minuscules.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- `market:` — AIR envoie les requêtes `market:` à l'application Market généralement prise en charge sur les périphériques Android.

```
navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
```

Lorsque le système d'exploitation le permet, les applications peuvent définir et enregistrer des modèles d'URI personnalisés. Vous pouvez créer une URL à l'aide du modèle pour lancer l'application à partir d'AIR.

Sandbox distants

Les modèles suivants sont autorisés. Utilisez-les comme dans un navigateur Web.

- `http:`
- `https:`
- `mailto:` AIR dirige ces requêtes à l'application de messagerie système enregistrée

Tous les autres modèles d'URI sont interdits.

Sandbox local avec fichiers

Les modèles suivants sont autorisés. Utilisez-les comme dans un navigateur Web.

- `file:`
- `mailto:` AIR dirige ces requêtes à l'application de messagerie système enregistrée

Tous les autres modèles d'URI sont interdits.

Sandbox local avec accès au réseau

Les modèles suivants sont autorisés. Utilisez-les comme dans un navigateur Web.

- `http:`
- `https:`
- `mailto:` AIR dirige ces requêtes à l'application de messagerie système enregistrée

Tous les autres modèles d'URI sont interdits.

Sandbox approuvé localement

Les modèles suivants sont autorisés. Utilisez-les comme dans un navigateur Web.

- `file:`
- `http:`
- https:
- `mailto:` AIR dirige ces requêtes à l'application de messagerie système enregistrée

Tous les autres modèles d'URI sont interdits.

Chapitre 45 : Communications avec d'autres occurrences de Flash Player et d'AIR

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe LocalConnection assure les communications entre les applications Adobe® AIR®, ainsi qu'entre les contenus SWF qui s'exécutent dans le navigateur. Vous disposez également de la classe LocalConnection pour communiquer entre une application AIR et un contenu SWF qui s'exécute dans le navigateur. La classe LocalConnection permet de développer des applications particulièrement versatiles capables de partager des données entre occurrences de Flash Player et d'AIR.

A propos de la classe LocalConnection

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe LocalConnection permet de développer des fichiers SWF qui peuvent échanger des instructions avec d'autres fichiers SWF sans utiliser la méthode fscommand() ou JavaScript. Les objets LocalConnection peuvent uniquement communiquer entre des fichiers SWF exécutés sur le même ordinateur client, mais ils peuvent concerner différentes applications. Par exemple, un fichier SWF exécuté dans un navigateur et un fichier SWF de projection peuvent partager des informations, le fichier de projection se chargeant de maintenir les informations locales et le fichier SWF du navigateur se connectant à distance. (un fichier de projection est un fichier SWF enregistré dans un format tel qu'il peut s'exécuter de manière autonome ; il n'est donc pas nécessaire de disposer de Flash Player pour le lire, car il est incorporé dans le fichier exécutable).

Les objets LocalConnection peuvent être utilisés pour communiquer entre des SWF utilisant différentes versions d'ActionScript :

- Les objets LocalConnection créés dans ActionScript 1.0 ou 2.0 peuvent communiquer avec les objets LocalConnection créés dans ActionScript 3.0.
- Les objets LocalConnection créés dans ActionScript 1.0 ou 2.0 peuvent communiquer avec les objets LocalConnection créés dans ActionScript 3.0.

Flash Player gère automatiquement les communications entre les objets LocalConnection de versions différentes.

La façon la plus simple d'utiliser un objet LocalConnection consiste à autoriser la communication uniquement entre des objets LocalConnection se trouvant dans le même domaine ou dans la même application AIR. Vous évitez ainsi les problèmes de sécurité. Toutefois, si vous devez autoriser les communications entre les domaines, vous disposez de diverses méthodes pour mettre en œuvre des mesures de sécurité. Pour plus d'informations, voir l'analyse du paramètre `connectionName` de la méthode `send()`, ainsi que les entrées `allowDomain()` et `domain` dans le code associé à la classe LocalConnection du manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Il est possible d'utiliser des objets LocalConnection pour envoyer et recevoir des données au sein d'un fichier SWF unique. Toutefois, Adobe ne recommande pas cette pratique. Faites de préférence appel aux objets partagés.

Les méthodes de rappel peuvent être ajoutées aux objets LocalConnection de trois manières :

- Créer des sous-classes de LocalConnection et ajouter des méthodes
- Définir la propriété `LocalConnection.client` sur un objet qui implémente ces méthodes
- Créer une classe dynamique qui étend la classe LocalConnection et y joindre dynamiquement des méthodes

La première manière d'ajouter des méthodes de rappel est d'étendre la classe LocalConnection. Vous pouvez définir les méthodes au sein de la classe personnalisée, plutôt que de les ajouter dynamiquement à l'occurrence de LocalConnection. Ceci est illustré par le code suivant :.

```
package
{
    import flash.net.LocalConnection;
    public class CustomLocalConnection extends LocalConnection
    {
        public function CustomLocalConnection(connectionName:String)
        {
            try
            {
                connect(connectionName);
            }
            catch (error:ArgumentError)
            {
                // server already created/connected
            }
        }
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

Pour créer une occurrence de la classe CustomLocalConnection, vous pouvez utiliser le code suivant :

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

La deuxième manière d'ajouter des méthodes de rappel consiste à utiliser la propriété `LocalConnection.client`. Il s'agit de créer une classe personnalisée et d'affecter une nouvelle occurrence à la propriété `client`, comme le montre le code suivant :

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

La propriété `LocalConnection.client` indique les méthodes de rappel d'objet à appeler. Dans le code précédent la propriété `client` était définie sur une nouvelle occurrence de la classe personnalisée CustomClient. La valeur par défaut de la propriété `client` est l'occurrence de LocalConnection actuelle. Vous pouvez utiliser la propriété `client` si vous disposez de deux gestionnaires de données dotés du même jeu de méthodes mais qui agissent différemment ; par exemple, dans une application où un bouton situé dans une fenêtre permet d'afficher le contenu d'une deuxième fenêtre.

Pour créer la classe CustomClient, vous pouvez utiliser le code suivant :

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

La troisième manière d'ajouter des méthodes de rappel consiste à créer une classe dynamique et d'y joindre dynamiquement des méthodes. Elle se rapproche de l'utilisation de la classe LocalConnection dans les versions précédentes d'ActionScript, comme le montre le code suivant :

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

Les méthodes de rappel peuvent être ajoutées dynamiquement à cette classe à l'aide du code suivant :

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

Cette manière d'ajouter des méthodes de rappel est déconseillée, car le code n'est pas vraiment portable. En outre, cette méthode de création de connexions locales pourrait présenter des problèmes de performance car l'accès aux propriétés dynamiques prend bien plus de temps que l'accès aux propriétés scellées.

Propriété isPerUser

La propriété `isPerUser` a été intégrée à Flash Player (10.0.32) et AIR (1.5.2) pour résoudre un conflit qui se produit lorsque plusieurs utilisateurs ont ouvert une session sur un ordinateur Mac. Elle n'est pas prise en charge par les autres systèmes d'exploitation, car seuls des utilisateurs spécifiques sont autorisés à se connecter localement. Définissez la propriété `isPerUser` sur `true` dans le nouveau code. Actuellement, la valeur par défaut correspond toutefois à `false` à des fins de compatibilité ascendante. Elle sera peut-être modifiée dans les versions futures des moteurs d'exécution.

Envoi de messages entre deux applications

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser la classe LocalConnection pour communiquer d'une part entre différentes applications AIR et, d'autre part, entre différentes applications Adobe® Flash® Player (SWF) qui s'exécutent dans un navigateur. Vous disposez également de la classe LocalConnection pour communiquer entre une application AIR et une application SWF qui s'exécute dans un navigateur.

Par exemple, vous pouvez utiliser plusieurs occurrences de Flash Player sur une même page Web ou vous servir d'une occurrence de Flash Player pour récupérer des données dans une occurrence de Flash Player affichée dans une fenêtre distincte.

Le code ci-dessous définit un objet LocalConnection qui joue le rôle d'un serveur et qui accepte des appels LocalConnection provenant d'autres applications :

```
package
{
    import flash.net.LocalConnection;
    import flash.display.Sprite;
    public class ServerLC extends Sprite
    {
        public function ServerLC()
        {
            var lc:LocalConnection = new LocalConnection();
            lc.client = new CustomClient1();
            try
            {
                lc.connect("conn1");
            }
            catch (error:Error)
            {
                trace("error:: already connected");
            }
        }
    }
}
```

Ce code crée un objet LocalConnection appelé `lc` et définit la propriété `client` sur une classe `clientObject`. Lorsqu'une autre application appelle une méthode dans cette occurrence de LocalConnection, le moteur d'exécution recherche cette méthode dans l'objet `clientObject`.

Si une connexion portant le nom spécifié est déjà établie, une exception Argument Error est renvoyée, ce qui indique que la tentative de connexion a échoué parce que l'objet était déjà connecté.

Dès qu'une occurrence de Flash Player se connecte à ce fichier SWF et essaie d'appeler l'une des méthodes de la connexion local, la requête est envoyée à la classe spécifiée par la propriété `client`, à savoir la classe CustomClient1 :

```
package
{
    import flash.events.*;
    import flash.system.fscommand;
    import flash.utils.Timer;
    public class CustomClient1 extends Object
    {
        public function doMessage(value:String = ""):void
        {
            trace(value);
        }
        public function doQuit():void
        {
            trace("quitting in 5 seconds");
            this.close();
            var quitTimer:Timer = new Timer(5000, 1);
            quitTimer.addEventListener(TimerEvent.TIMER, closeHandler);
        }
        public function closeHandler(event:TimerEvent):void
        {
            fscommand("quit");
        }
    }
}
```

Pour créer un serveur LocalConnection, appelez la méthode `LocalConnection.connect()` et fournissez un nom de connexion unique. Si une connexion est déjà ouverte avec le nom choisi, une erreur ArgumentError est générée, ce qui indique que la tentative de connexion a échoué parce que l'objet était déjà connecté.

L'extrait de code suivant illustre comment créer une LocalConnection appelée `conn1` :

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

La connexion à l'application principale à partir d'une application secondaire nécessite que vous créiez d'abord un objet LocalConnection dans l'objet LocalConnection d'envoi, puis que vous appeliez la méthode `LocalConnection.send()` avec le nom de la connexion et le nom de la méthode à exécuter. Par exemple, pour envoyer la méthode `doQuit` à l'objet LocalConnection créé précédemment, utilisez le code suivant :

```
sendingConnection.send("conn1", "doQuit");
```

Ce code établit une connexion `conn1` à l'objet LocalConnection existant, puis appelle la méthode `doMessage()` dans l'application distante. Si vous souhaitez envoyer des paramètres à l'application distante, spécifiez les arguments supplémentaires après le nom de la méthode `send()`, comme le montre l'extrait de code suivant :

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

Connexion au contenu dans des domaines différents et aux applications AIR

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour autoriser uniquement les communications issues de domaines précis, appelez la méthode `allowDomain()` ou `allowInsecureDomain()` de la classe LocalConnection et transmettez la liste des domaines autorisés à accéder à cet objet LocalConnection.

Dans les versions antérieures d'ActionScript, `LocalConnection.allowDomain()` et `LocalConnection.allowInsecureDomain()` étaient des méthodes de rappel que les développeurs devaient implémenter et qui devaient renvoyer une valeur booléenne. Dans ActionScript 3.0, `LocalConnection.allowDomain()` et `LocalConnection.allowInsecureDomain()` sont deux méthodes intégrées, que les développeurs peuvent appeler de la même façon que `Security.allowDomain()` et `Security.allowInsecureDomain()`, en transmettant un ou plusieurs noms de domaines à autoriser.

Dans Flash Player 8, des restrictions de sécurité relatives aux fichiers SWF locaux ont été introduites. Un fichier SWF autorisé à accéder à Internet n'a pas accès au système de fichiers local. Si vous spécifiez `localhost`, tout fichier SWF local peut accéder à ce fichier SWF. Si la méthode `LocalConnection.send()` tente de communiquer avec un fichier SWF à partir d'un sandbox de sécurité auquel le code d'appel n'a pas accès, un événement `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) est distribué. Pour contourner cette erreur, vous pouvez spécifier le domaine de l'appelant dans la méthode `LocalConnection.allowDomain()` du destinataire.

Deux valeurs spéciales peuvent être transmises aux méthodes `LocalConnection.allowDomain()` et `LocalConnection.allowInsecureDomain()` : `*` et `localhost`. L'astérisque `(*)` permet un accès à partir de tous les domaines. La chaîne `localhost` autorise les appels à l'application à partir du contenu installé localement mais hors du répertoire des ressources de l'application.

Si la méthode `LocalConnection.send()` essaie de communiquer avec une application à partir d'un sandbox de sécurité auquel le code d'appel n'a pas accès, un événement `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) est distribué. Pour contourner cette erreur, vous pouvez spécifier le domaine de l'appelant dans la méthode `LocalConnection.allowDomain()` du destinataire.

Si vous mettez en œuvre la communication uniquement entre contenus du même domaine, vous pouvez spécifier un paramètre `connectionName` qui ne commence pas par un caractère de soulignement (_) et ne spécifie pas un nom de domaine (par exemple `myDomain:connectionName`). Utilisez la même chaîne dans la commande `LocalConnection.connect(connectionName)`.

Si vous mettez en œuvre la communication entre contenus de domaines différents, spécifiez un paramètre `connectionName` qui commence par un caractère de soulignement. L'ajout d'un caractère de soulignement améliore la portabilité entre domaines du contenu qui possède l'objet LocalConnection de réception. Les deux cas de figure possibles sont les suivants :

- Si la chaîne associée à `connectionName` ne commence pas par un caractère de soulignement, le moteur d'exécution ajoute un préfixe au nom du superdomaine suivi de deux points, comme dans `myDomain:connectionName`. Vous avez ainsi la garantie que votre connexion n'entrera pas en conflit avec les connexions de même nom dans d'autres domaines. Mais tous les objets LocalConnection d'envoi doivent spécifier ce superdomaine, comme dans `myDomain:connectionName`. Si le fichier HTML ou SWF associé à l'objet LocalConnection de réception est déplacé vers un autre domaine, le moteur d'exécution modifie le préfixe afin qu'il reflète le nouveau superdomaine, comme dans `anotherDomain:connectionName`. Tous les objets LocalConnection d'envoi doivent être modifiés manuellement pour pointer vers le nouveau superdomaine.
- Si la chaîne associée à `connectionName` commence par un caractère de soulignement, comme dans `_connectionName`), le moteur d'exécution ne lui ajoute pas de préfixe. Les objets LocalConnection de réception et d'envoi utilisent donc des chaînes identiques pour `connectionName`. Si l'objet de réception utilise `LocalConnection.allowDomain()` pour spécifier que les connexions seront acceptées à partir de tous les domaines, le fichier HTML ou SWF contenant l'objet LocalConnection de réception peut être déplacé vers un autre domaine, sans qu'il soit nécessaire de modifier les objets LocalConnection d'envoi.

  Il existe un inconvénient à utiliser des noms avec caractère de soulignement dans `connectionName` : c'est le risque de collision. Deux applications peuvent tenter de se connecter à l'aide du même `connectionName`. Il en existe un deuxième lié à la sécurité. Les noms de connexion qui utilisent la syntaxe du caractère de soulignement n'identifient pas le domaine de l'application à l'écoute. C'est pour ces raisons qu'il est préférable de recourir à des noms dotés de qualificatifs de domaines.

Adobe AIR

Pour communiquer avec le contenu qui s'exécute dans le sandbox de sécurité de l'application AIR (contenu installé avec l'application AIR), vous devez insérer avant le nom de connexion un superdomaine qui identifie l'application AIR. La chaîne du superdomaine commence par `app#`, suivi de l'identifiant d'application, suivi d'un point (.), suivi de l'identifiant d'éditeur (s'il est défini). Ainsi, le superdomaine adapté au paramètre `connectionName` si l'identifiant d'application est `com.example.air.MyApp` et qu'aucun identifiant d'éditeur n'est stipulé, correspond à : `« app#com.example.air.MyApp »`. Par conséquent, si le nom de la connexion de base est « appConnection », la chaîne entière à utiliser dans le paramètre `connectionName` correspond à :
`« app#com.example.air.MyApp:appConnection »`. Si l'application stipule l'identifiant d'éditeur, ce dernier doit être inclus dans la chaîne du superdomaine, comme suit :
`« app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1 »`.

Lorsque vous autorisez une autre application AIR à communiquer avec votre application via la connexion locale, vous devez appeler l'élément `allowDomain()` de l'objet LocalConnection en transmettant le nom du domaine de la connexion locale. Pour une application AIR, ce nom de domaine est formé à partir des ID d'application et d'éditeur, de la même façon que la chaîne de connexion. Par exemple, si l'application AIR d'envoi a pour ID d'application `com.example.air.FriendlyApp` et pour ID d'éditeur `214649436BD677B62C33D02233043EA236D13934.1`, la chaîne de domaine que vous utiliseriez pour autoriser cette application à se connecter est : `app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1`. (Depuis la version 1.5.3 d'AIR, certaines applications AIR ne possèdent pas d'identifiant d'éditeur.)

Chapitre 46 : Communication avec les processus natifs dans AIR

Adobe AIR 2 et ultérieur

Depuis Adobe AIR 2, les applications AIR peuvent exécuter d'autres processus natifs et communiquer avec eux via la ligne de commande. Une application AIR est ainsi capable d'exécuter un processus et de communiquer avec lui via les flux d'entrée et de sortie standard.

Pour communiquer avec les processus natifs, mettez en package une application AIR à installer via un programme d'installation natif. Le type de fichier du programme d'installation natif varie selon le système d'exploitation pour lequel il est créé :

- C'est un fichier DMG sous Mac OS.
- C'est un fichier EXE sous Windows.
- C'est un package RPM ou DEB sous Linux.

Ces applications portent le nom d'applications à profil de bureau étendu. Pour créer un fichier de programme d'installation natif, vous spécifiez l'option `-target native` lorsque vous appelez la commande `-package` à l'aide d'ADT.

Voir aussi

flash.filesystem.File.openWithDefaultApplication()

flash.desktop.NativeProcess

Présentation des communications avec les processus natifs

Adobe AIR 2 et ultérieur

Une application AIR figurant dans le profil de bureau étendu peut exécuter un fichier comme s'il était appelé par la ligne de commande. Elle peut communiquer avec les flux standard du processus natif. Les flux standard incluent le flux d'entrée standard (stdin), le flux de sortie (stdout) et le flux d'erreur standard (stderr).

Remarque : *les applications dotées d'un profil de bureau étendu peuvent également lancer des fichiers et des applications par le biais de la méthode* `File.openWithDefaultApplication()`*. L'utilisation de cette méthode ne permet toutefois pas à l'application AIR d'accéder aux flux standard. Pour plus d'informations, voir «* *Ouverture de fichiers dans l'application système définie par défaut » à la page 706*

Le code suivant illustre le lancement d'une application test.exe dans le répertoire d'application. L'application transmet l'argument `"hello"` sous forme d'argument de ligne de commande et ajoute un écouteur d'événements au flux de sortie standard du processus :

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
var processArgs:Vector.<String> = new Vector.<String>();
processArgs.push("hello");
nativeProcessStartupInfo.arguments = processArgs;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_OUTPUT_DATA, onOutputData);
process.start(nativeProcessStartupInfo);
public function onOutputData(event:ProgressEvent):void
{
    var stdOut:ByteArray = process.standardOutput;
    var data:String = stdOut.readUTFBytes(process.standardOutput.bytesAvailable);
    trace("Got: ", data);
}
```

Lancement et fermeture d'un processus natif

Adobe AIR 2 et ultérieur

Pour lancer un processus natif, définissez un objet NativeProcessInfo comme suit :

- Pointez vers le fichier à lancer.
- Enregistrez les arguments de ligne de commande à transmettre au processus lors du lancement (facultatif).
- Définissez le répertoire de travail du processus (facultatif).

Pour démarrer le processus natif, transmettez l'objet NativeProcessInfo en tant que paramètre de la méthode `start()` d'un objet NativeProcess.

Le code suivant illustre par exemple le lancement d'une application test.exe dans le répertoire d'application. L'application transmet l'argument `"hello"` et définit le répertoire documents de l'utilisateur en tant que répertoire de travail :

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
var processArgs:Vector.<String> = new Vector.<String>();
processArgs[0] = "hello";
nativeProcessStartupInfo.arguments = processArgs;
nativeProcessStartupInfo.workingDirectory = File.documentsDirectory;
process = new NativeProcess();
process.start(nativeProcessStartupInfo);
```

Pour mettre fin au processus, appelez la méthode `exit()` de l'objet NativeProcess.

Si vous souhaitez qu'un fichier puisse être exécuté dans l'application installée, assurez-vous qu'il peut être exécuté sur le système de fichiers lors de la mise en package de l'application. (Sous Mac et Linux, vous disposez de chmod pour définir l'indicateur executable, le cas échéant.)

Communications avec un processus natif

Adobe AIR 2 et ultérieur

Lorsqu'une application AIR démarre un processus natif, elle peut communiquer avec les flux d'entrée, de sortie et d'erreur standard de ce dernier.

Vous lisez et écrivez les données dans les flux à l'aide des propriétés suivantes de l'objet NativeProcess :

- `standardInput` : permet d'accéder aux données du flux d'entrée standard.
- `standardOutput` : permet d'accéder aux données du flux de sortie standard.
- `standardError` : permet d'accéder aux données du flux d'erreur standard.

Ecriture dans le flux d'entrée standard

Vous pouvez écrire des données dans le flux d'entrée standard à l'aide des méthodes d'écriture de la propriété `standardInput` de l'objet NativeProcess. Lorsque l'application AIR écrit des données dans le processus, l'objet NativeProcess distribue des événements `standardInputProgress`.

S'il se produit une erreur lors de l'écriture dans le flux d'entrée standard, l'objet NativeProcess distribue un événement `ioErrorStandardInput`.

Pour fermer le flux d'entrée, appelez la méthode `closeInput()` de l'objet NativeProcess. Lors de la fermeture du flux d'entrée, l'objet NativeProcess distribue un événement `standardInputClose`.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.start(nativeProcessStartupInfo);
process.standardInput.writeUTF("foo");
if(process.running)
{
    process.closeInput();
}
```

Lecture dans le flux de sortie standard

Vous pouvez lire les données du flux de sortie standard par le biais des méthodes de lecture de cette propriété. Au fur et à mesure que l'application AIR extrait du processus les données du flux de sortie, l'objet NativeProcess distribue des événements `standardOutputData`.

S'il se produit une erreur lors de l'écriture du flux de sortie standard, l'objet NativeProcess distribue un événement `standardOutputError`.

Lorsque le processus ferme le flux de sortie, l'objet NativeProcess distribue un événement `standardOutputClose`.

Veillez à lire les données issues du flux d'entrée standard au fur et à mesure de leur génération. En d'autres termes, associez un écouteur à l'événement `standardOutputData`. Dans l'écouteur d'événements `standardOutputData`, lisez les données issues de la propriété `standardOutput` de l'objet NativeProcess. N'attendez pas l'événement `standardOutputClose` ou `exit` pour lire l'ensemble des données. Si vous ne lisez pas les données au fur et à mesure que le processus natif les génère, la mémoire tampon risque d'être saturée ou les données de se perdre. Une mémoire tampon saturée risque d'entraîner le blocage du processus natif lorsqu'il essaie d'écrire d'autres données. Toutefois, si vous n'associez pas d'écouteur à l'événement `standardOutputData`, la mémoire tampon ne se remplit pas et le processus ne se bloque pas. Il vous sera dans ce cas impossible d'accéder aux données.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_OUTPUT_DATA, dataHandler);
process.start(nativeProcessStartupInfo);
var bytes:ByteArray = new ByteArray();
function dataHandler(event:ProgressEvent):void
{
    bytes.writeBytes(process.standardOutput.readBytes(process.standardOutput.bytesAvailable);
}
```

Lecture dans le flux d'erreur standard

Vous pouvez lire les données du flux d'erreur standard par le biais des méthodes de lecture de cette propriété. Au fur et à mesure que l'application AIR lit dans le processus les données du flux d'erreur, l'objet NativeProcess distribue des événements `standardErrorData`.

S'il se produit une erreur lors de l'écriture du flux d'erreur standard, l'objet NativeProcess distribue un événement `standardErrorIoError`.

Lorsque le processus ferme le flux d'erreur, l'objet NativeProcess distribue un événement `standardErrorClose`..

Veillez à lire les données issues du flux d'erreur standard au fur et à mesure de leur génération. En d'autres termes, associez un écouteur à l'événement `standardErrorData`. Dans l'écouteur d'événements `standardErrorData`, lisez les données extraites de la propriété `standardError` de l'objet NativeProcess. N'attendez pas l'événement `standardErrorClose` ou `exit` pour lire l'ensemble des données. Si vous ne lisez pas les données au fur et à mesure que le processus natif les génère, la mémoire tampon risque d'être saturée ou les données de se perdre. Une mémoire tampon saturée risque d'entraîner le blocage du processus natif lorsqu'il essaie d'écrire d'autres données. Toutefois, si vous n'associez pas d'écouteur à l'événement `standardErrorData`, la mémoire tampon ne se remplit pas et le processus ne se bloque pas. Il vous sera dans ce cas impossible d'accéder aux données.

```
var nativeProcessStartupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
var file:File = File.applicationDirectory.resolvePath("test.exe");
nativeProcessStartupInfo.executable = file;
process = new NativeProcess();
process.addEventListener(ProgressEvent.STANDARD_ERROR_DATA, errorDataHandler);
process.start(nativeProcessStartupInfo);
var errorBytes:ByteArray = new ByteArray();
function errorDataHandler(event:ProgressEvent):void
{
    bytes.writeBytes(process.standardError.readBytes(process.standardError.bytesAvailable);
}
```

Considérations liées à la sécurité qui affectent les communications avec le processus natif

Adobe AIR 2 et ultérieur

L'API de processus natif peut exécuter tout fichier exécutable sur le système de l'utilisateur. Soyez extrêmement attentif lors de la création et de l'exécution de commandes. Si toute partie d'une commande à exécuter est issue d'une source externe, vérifiez méticuleusement son intégrité avant de l'exécuter. L'application AIR doit de même valider toute donnée transmise à un processus en cours d'exécution.

Toutefois, la validation d'une entrée s'avère parfois complexe. Par parer à ce problème, il est recommandé de programmer une application native (un fichier EXE sous Windows, par exemple) qui contient des API déterminées. Ces API ne doivent traiter que les commandes définies par l'application. L'application peut ainsi n'accepter qu'un jeu réduit d'instructions via le flux d'entrée standard.

Sous Windows, AIR ne vous autorise pas à exécuter directement les fichiers .bat. L'interpréteur de commande (l'application cmd.exe) exécute les fichiers .bat Windows. Lorsque vous appelez un fichier .bat, cette application peut interpréter les arguments transmis à la commande en tant applications supplémentaires à lancer. L'insertion malveillante de caractères supplémentaires dans la chaîne d'arguments risque de mener cmd.exe à exécuter une application nuisible ou non sécurisée. Ainsi, sans validation correcte des données, l'application AIR risque d'appeler `myBat.bat myArguments c:/evil.exe`. Outre le fichier batch, l'interpréteur de commandes lancerait l'application evil.exe.

Chapitre 47 : Utilisation de l'API externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API externe ActionScript 3.0 (flash.external.ExternalInterface) autorise les communications simples entre ActionScript et l'application conteneur dans laquelle s'exécute Adobe Flash Player. Utilisez l'API ExternalInterface pour créer une interaction entre un document SWF et JavaScript dans une page HTML.

Vous pouvez utiliser l'API externe pour interagir avec une application conteneur et transmettre les données entre ActionScript et JavaScript dans une page HTML.

Exemples de tâches courantes de l'API externe :

- Obtention d'informations sur l'application conteneur
- Utilisation d'ActionScript pour appeler le code dans une page Web affichée dans un navigateur ou une application de bureau AIR
- Appel du code ActionScript depuis une page Web
- Création d'un proxy pour simplifier l'appel de code ActionScript depuis une page Web

Remarque : *seule une communication entre ActionScript dans un fichier SWF et l'application conteneur qui comprend une référence à Flash Player ou à l'occurrence dans laquelle ce fichier SWF est chargé est passée en revue ci-après. Toute autre utilisation de Flash Player dans une application n'est pas traitée dans cette documentation. Flash Player est conçu pour être utilisé comme plug-in de navigation ou de projection (application autonome). Les autres scénarios d'utilisation peuvent avoir une prise en charge limitée.*

Utilisation de l'API externe dans AIR

Etant donné qu'une application AIR ne possède pas de conteneur externe, l'interface externe n'est généralement ni utilisée, ni requise. Lorsque l'application AIR charge directement un fichier SWF, le code correspondant peut communiquer directement avec le code ActionScript dans le fichier SWF (conformément aux restrictions du sandbox de sécurité).

Toutefois, lorsque l'application AIR charge un fichier SWF par le biais d'une page HTML dans un objet HTMLLoader (ou un composant HTML dans Flex), cet objet sert de conteneur externe. Le code ActionScript figurant dans le fichier SWF chargé peut ainsi communiquer avec le code JavaScript figurant dans la page HTML chargée dans l'objet HTMLLoader, par le biais de l'interface externe.

Principes de base de l'utilisation de l'API externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Même si, dans certains cas, un fichier SWF peut s'exécuter tout seul (si vous faites appel à Adobe® Flash® Professional pour créer un fichier de projection SWF, par exemple), une application SWF s'exécute généralement comme un élément intégré à une autre application. De façon générale, le conteneur dans lequel le fichier SWF est intégré est un fichier HTML ; un fichier SWF est utilisé, moins fréquemment, pour une partie ou l'intégralité de l'interface utilisateur d'une application de bureau.

Au fur et à mesure que vous utilisez des applications plus avancées, il se peut que vous souhaitiez définir une communication entre le fichier SWF et l'application du conteneur. Par exemple, il est courant pour une page Web d'afficher du texte ou d'autres informations en HTML, et d'inclure un fichier SWF pour afficher du contenu visuel dynamique (diagramme ou vidéo). Dans ce cas, vous souhaitez peut-être que lorsque les utilisateurs cliquent sur un bouton de la page Web, le fichier SWF soit modifié. ActionScript contient un mécanisme connu sous le nom d'API externe, qui facilite ce type de communication entre ActionScript dans un fichier SWF et d'autre code dans l'application conteneur.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à cette fonctionnalité.

Application conteneur Application au sein de laquelle Flash Player exécute un fichier SWF, notamment un navigateur Web et une page HTML incluant le contenu Flash Player ou une application AIR qui charge le fichier SWF dans une page Web.

Fichier de projection Fichier exécutable qui comprend un contenu SWF et une version intégrée de Flash Player. Pour créer un fichier de projection, faites appel à Flash Professional ou à la version autonome de Flash Player. Les fichiers de projection sont généralement utilisés pour distribuer des fichiers SWF par CD-ROM ou dans des situations semblables, lorsque le volume à télécharger n'est pas un problème et que l'auteur du SWF souhaite être sûr que l'utilisateur pourra exécuter le fichier SWF, indépendamment du fait que Flash Player est installé sur l'ordinateur de l'utilisateur.

Proxy Code ou application intermédiaire qui appelle du code dans une application (l'application externe) au nom d'une autre application (l'application appelante), et renvoie des valeurs à l'application appelante. Un proxy peut être utilisé pour différentes raisons, notamment :

- Pour simplifier le processus d'exécution d'appels de fonction externe en convertissant des appels de fonction native dans l'application appelante au format compris par l'application externe.
- Pour contourner les limites de sécurité ou autres qui empêchent l'appelant de communiquer directement avec l'application externe.

Sérialiser Convertir des objets ou des valeurs de données en un format qui permet de transmettre les valeurs dans des messages entre deux systèmes de programmation (par exemple, sur Internet ou entre deux applications qui s'exécutent sur un seul ordinateur).

Utilisation des exemples

La plupart des exemples sont des blocs de code de petite taille proposés à des fins de démonstration, plutôt que des exemples complets ou des codes qui vérifient des valeurs. Etant donné que l'utilisation de l'API externe exige (par définition) l'écriture de code ActionScript ainsi que de code dans une application conteneur, le test des exemples implique la création d'un conteneur (par exemple, une page Web contenant le fichier SWF) et l'utilisation des codes pour interagir avec le conteneur.

Pour tester un exemple de communication entre ActionScript et JavaScript :

1. Créez un document vide à l'aide de Flash Professional et enregistrez-le dans votre ordinateur.
2. Dans le menu principal, choisissez Fichier > Paramètres de publication.
3. Dans l'onglet Formats de la boîte de dialogue Paramètres de publication, vérifiez que les cases à cocher Flash et HTML sont sélectionnées.
4. Cliquez sur le bouton Publier. Ceci génère un fichier SWF et un fichier HTML dans le même dossier et avec le même nom que vous avez utilisé pour enregistrer le document Fermez la boîte de dialogue Paramètres de publication en cliquant sur le bouton OK.

5 Désélectionnez la case à cocher HTML. Une fois que la page HTML est générée, vous pouvez la modifier pour ajouter le code JavaScript approprié. Lorsque vous désactivez l'option HTML, vous vous assurez qu'après avoir modifié la page HTML, Flash n'écrasera pas vos changements avec une nouvelle page HTML lors de la publication du fichier SWF.

6 Fermez la boîte de dialogue Paramètres de publication en cliquant sur le bouton OK.

7 Avec une application d'éditeur de texte ou HTML, ouvrez le fichier HTML créé par Flash lorsque vous avez publié le fichier SWF. Dans le code source HTML, ajoutez des balises `script` d'ouverture et de fermeture et copiez-y le code JavaScript issu de l'exemple de code :

```
<script>
// add the sample JavaScript code here
</script>
```

8 Enregistrez le fichier HTML et revenez à Flash.

9 Sélectionnez l'image-clé sur l'image 1 du scénario puis ouvrez le panneau Actions.

10 Copiez le code ActionScript dans le panneau Script.

11 Dans le menu principal, choisissez Fichier > Publier pour mettre à jour le fichier SWF avec les changements que vous avez effectués.

12 Utilisez un navigateur Web pour ouvrir la page HTML que vous avez modifiée afin d'afficher la page et de tester la communication entre ActionScript et la page HTML.

Pour tester un exemple de communication de conteneur ActionScript-vers-ActiveX :

1 Créez un document vide à l'aide de Flash Professional et enregistrez-le dans votre ordinateur. Vous pouvez l'enregistrer dans le dossier dans lequel votre application conteneur s'attend à trouver le fichier SWF.

2 Dans le menu principal, choisissez Fichier > Paramètres de publication.

3 Dans l'onglet Formats de la boîte de dialogue Paramètres de publication, vérifiez que seule la case à cocher Flash est sélectionnée.

4 Dans le champ Fichier situé en regard de la case à cocher Flash, cliquez sur l'icône de dossier pour sélectionner le dossier dans lequel votre fichier SWF sera publié. Définissez l'emplacement de votre fichier SWF pour pouvoir (par exemple) conserver le document dans un dossier, mais placer le fichier SWF publié dans un autre (le dossier contenant le code source pour l'application conteneur, par exemple).

5 Sélectionnez l'image-clé sur l'image 1 du scénario puis ouvrez le panneau Actions.

6 Copiez le code ActionScript pour l'exemple dans le panneau Script.

7 Dans le menu principal, choisissez Fichier > Publier pour publier de nouveau le fichier SWF.

8 Créez et exécutez votre application conteneur afin de tester la communication entre elle et ActionScript.

Pour obtenir des exemples complets d'utilisation de l'API externe pour communiquer avec une page HTML, voir le sujet suivant :

- « Exemple d'API externe : communications entre ActionScript et JavaScript dans un navigateur Web » à la page 882

Ces exemples comprennent le code entier (y compris le code ActionScript et de vérification des erreurs du conteneur) que vous devez utiliser lorsque vous écrivez le code à l'aide de l'API externe. Pour consulter un autre exemple complet d'utilisation de l'API externe, voir l'exemple associé à la classe ExternalInterface dans le Guide de référence ActionScript 3.0 pour Flash Professional.

Avantages de l'API externe et conditions requises

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API externe correspond à la partie d'ActionScript qui fournit le mécanisme de communication entre ActionScript et le code exécuté dans une application dite externe, c'est-à-dire qui joue le rôle de conteneur pour Flash Player (en général un navigateur Web ou une application de projection autonome). Dans ActionScript 3.0, la fonctionnalité de l'API externe est assurée par la classe ExternalInterface. Dans les versions de Flash Player antérieures à Flash Player 8, l'action `fscommand()` était utilisée pour établir les communications avec l'application conteneur. La classe ExternalInterface remplace l'action `fscommand()`.

Remarque : *si vous devez utiliser l'ancienne fonction `fscommand()` (par exemple pour maintenir la compatibilité avec d'anciennes applications ou pour interagir avec une application conteneur SWF tierce ou avec le lecteur autonome Flash Player), elle est disponible au niveau du package flash.system.*

La classe ExternalInterface est un sous-système permettant de communiquer facilement d'ActionScript et Flash Player à JavaScript dans une page HTML.

La classe ExternalInterface est disponible uniquement dans les circonstances suivantes :

- Dans toutes les versions prises en charge d'Internet Explorer pour Windows (5.0 et les versions ultérieures)
- Dans un navigateur qui prend en charge l'interface NPRuntime, qui actuellement comprend Firefox 1.0 et versions ultérieures, Mozilla 1.7.5 et versions ultérieures, Netscape 8.0 et versions ultérieures, ainsi que Safari 1.3 et versions ultérieures.
- Dans une application AIR, lorsque le fichier SWF est intégré dans une page HTML affichée par la commande HTMLLoader.

Dans tous les autres cas de figure (par exemple exécution dans un lecteur autonome), la propriété `ExternalInterface.available` renvoie la valeur `false`.

Depuis ActionScript, vous pouvez appeler une fonction JavaScript sur la page HTML. L'API externe apporte les améliorations fonctionnelles suivantes grâce à `fscommand()` :

- Vous pouvez utiliser toutes les fonctions JavaScript, pas seulement les fonctions compatibles avec `fscommand()`.
- Vous pouvez transmettre n'importe quel nombre d'arguments, avec n'importe quels noms ; vous n'êtes pas limité à une commande et une chaîne d'argument. L'API externe offre donc bien plus de souplesse que `fscommand()`.
- Vous pouvez transmettre divers types de données (Boolean, Number et String, entre autres) et n'êtes plus limité aux paramètres String.
- Vous pouvez recevoir la valeur d'un appel, et cette valeur retourne immédiatement à ActionScript (comme la valeur de retour d'un appel que vous faites).

Important : *si le nom attribué à l'occurrence de Flash Player sur une page HTML (l'attribut `id` de la balise `object`) contient un tiret (-) ou un autre caractère défini en tant qu'opérateur dans JavaScript (tels `+`, `*`, `/`, `\`, `.`, etc.), les appels à ExternalInterface effectués depuis ActionScript ne fonctionnent pas lorsque la page Web du conteneur est consultée dans Internet Explorer. En outre, si les balises HTML qui définissent l'occurrence de Flash Player (les balises `object` et `embed`) sont imbriquées dans une balise `form` HTML, les appels à ExternalInterface effectués depuis ActionScript ne fonctionnent pas.*

Utilisation de la classe ExternalInterface

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La communication entre ActionScript et l'application conteneur peut se faire de deux façons : soit ActionScript appelle un élément de code (par exemple une fonction JavaScript) défini dans le conteneur, soit le code du conteneur appelle une fonction ActionScript définie comme pouvant être appelée. Dans les deux cas, des informations peuvent être envoyées au code appelé et les résultats renvoyés au code appelant.

Pour faciliter la communication, la classe ExternalInterface comprend deux propriétés statiques et deux méthodes statiques également. Ces propriétés et méthodes servent à obtenir des informations sur la connexion à l'interface externe, à exécuter le code dans le conteneur à partir d'ActionScript et de rendre disponibles les fonctions ActionScript que le conteneur doit appeler.

Obtention d'informations sur le conteneur externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété `ExternalInterface.available` indique si Flash Player se trouve dans un conteneur offrant une interface externe. Si l'interface externe est disponible, cette propriété est `true` ; sinon, elle est `false`. Avant d'utiliser toute autre fonctionnalité de la classe ExternalInterface, vous devez toujours vérifier que le conteneur actif prend en charge la communication avec l'interface externe, comme suit :

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

Remarque : *la propriété `ExternalInterface.available` indique si le conteneur actif prend en charge la connectivité ExternalInterface. Elle ne vous dit pas si JavaScript est activé dans le navigateur actif.*

La propriété `ExternalInterface.objectID` vous permet de déterminer l'identifiant unique de l'occurrence Flash Player (c'est-à-dire, l'attribut `id` de la balise `object` dans Internet Explorer ou l'attribut `name` de la balise `embed` dans les navigateurs utilisant l'interface NPRuntime). Cet identifiant unique représente le document SWF actif dans le navigateur et permet d'y faire référence, par exemple si vous appelez une fonction JavaScript dans une page HTML conteneur. Si le conteneur Flash Player n'est pas un navigateur Web, la propriété est `null`.

Appel de code externe à partir d'ActionScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `ExternalInterface.call()` exécute le code dans l'application conteneur. Elle nécessite au moins un paramètre, une chaîne contenant le nom de la fonction à appeler dans cette application. Tout paramètre supplémentaire transmis à la méthode `ExternalInterface.call()` est retransmis au conteneur comme paramètres de l'appel de fonction.

```
// calls the external function "addNumbers"
// passing two parameters, and assigning that function's result
// to the variable "result"
var param1:uint = 3;
var param2:uint = 7;
var result:uint = ExternalInterface.call("addNumbers", param1, param2);
```

Si le conteneur est une page HTML, la méthode appelle la fonction JavaScript avec le nom spécifié, qui doit être défini dans un élément `script` de la page HTML. La valeur de retour de la fonction JavaScript est retransmise à ActionScript.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

Si le conteneur est un autre conteneur ActiveX, le contrôle ActiveX Flash Player distribue alors son événement `FlashCall`. Flash Player sérialise le nom de fonction spécifié et les éventuels paramètres dans une chaîne XML. Le conteneur peut accéder à ces informations dans la propriété `request` de l'objet événement, puis les utiliser pour déterminer comment il doit exécuter son propre code. Pour renvoyer une valeur à ActionScript, le code conteneur appelle la méthode `SetReturnValue()` de l'objet ActiveX et transmet le résultat (sérialisé dans une chaîne XML) comme paramètre de cette méthode. Pour plus d'informations sur le format XML utilisé pour cette communication, voir « Format XML de l'API externe » à la page 880.

Que le conteneur soit un navigateur Web ou un autre conteneur ActiveX, si l'appel échoue ou que la méthode conteneur ne spécifie aucune valeur de retour, la valeur `null` est alors renvoyée. La méthode `ExternalInterface.call()` renvoie une exception SecurityError si l'environnement conteneur appartient à un sandbox de sécurité auquel le code appelant n'a pas accès. Vous pouvez contourner ce problème en attribuant à `allowScriptAccess` une valeur adaptée dans l'environnement conteneur. Par exemple, vous changez la valeur de `allowScriptAccess` dans une page HTML, vous devez modifier l'attribut approprié dans les balises `object` et `embed`.

Appel du code ActionScript à partir du conteneur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un conteneur peut uniquement appeler du code ActionScript compris dans une fonction, et aucun autre code ActionScript. Pour appeler une fonction ActionScript à partir de l'application conteneur, deux opérations sont nécessaires : enregistrer la fonction auprès de la classe ExternalInterface, puis l'appeler à partir du code du conteneur.

Dans un premier temps, vous devez enregistrer votre fonction ActionScript pour indiquer qu'elle doit être mise à disposition du conteneur. Pour ce faire, utilisez la méthode `ExternalInterface.addCallback()` comme suit :

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

La méthode `addCallback()` prend deux paramètres : le premier, un nom de fonction sous forme de chaîne, correspond au nom de la fonction pour le conteneur. Le second paramètre est la fonction ActionScript elle-même, qui doit s'exécuter lorsque le conteneur appelle le nom de fonction défini. Comme ces noms sont différents, le nom que le conteneur utilise peut être différent du véritable nom de la fonction ActionScript. Cela vous servira particulièrement si vous ne connaissez pas le nom de la fonction, par exemple si une fonction anonyme est spécifiée ou si la fonction à appeler est déterminée au moment de l'exécution.

Une fois que la fonction ActionScript a été enregistrée auprès de la classe ExternalInterface, le conteneur peut alors appeler la fonction. La procédure d'appel varie en fonction du type de conteneur. Par exemple, si le conteneur est du code JavaScript dans un navigateur, la fonction ActionScript est appelée avec le nom de fonction enregistré, comme s'il s'agissait d'une méthode de l'objet de navigateur Flash Player (c'est-à-dire une méthode de l'objet JavaScript représentant la balise `object` ou `embed`). En d'autres termes, les paramètres sont transmis et le résultat est renvoyé comme si une fonction locale était appelée.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

Si vous appelez une fonction ActionScript dans un fichier SWF exécuté dans une application de bureau, le nom de fonction enregistré et les éventuels paramètres doivent être sérialisés dans une chaîne au format XML. L'appel s'effectue alors sur la méthode `CallFunction()` du contrôle ActiveX avec comme paramètre la chaîne XML obtenue. Pour plus d'informations sur le format XML utilisé pour cette communication, voir « Format XML de l'API externe » à la page 880.

Dans les deux cas, la valeur de retour de la fonction ActionScript est retransmise au code du conteneur, directement sous forme de valeur si l'appelant est un code JavaScript dans un navigateur ou sous forme de chaîne XML s'il s'agit d'un conteneur ActiveX.

Format XML de l'API externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La communication entre ActionScript et une application hébergeant le contrôle Active X Shockwave Flash nécessite un format XML spécifique qui servira à convertir les appels de fonction et les valeurs. Le format XML utilisé par l'API externe se divise en deux parties. L'une est destinée à représenter les appels de fonction. L'autre sert à représenter des valeurs individuelles ; ce format s'applique aux paramètres des fonctions comme aux valeurs de retour. Le format XML des appels de fonction est utilisé pour les appels en provenance et à destination d'ActionScript. Pour un appel de fonction issu d'ActionScript, Flash Player transmet les informations XML au conteneur ; pour un appel provenant du conteneur, Flash Player attend de l'application une chaîne XML du même format. L'extrait XML suivant donne un exemple d'appel de fonction formaté :

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

Le nœud racine est le nœud `invoke`. Il possède deux attributs : `name` indique le nom de la fonction à appeler et `returntype` a toujours la valeur `xml`. Si l'appel de fonction inclut des paramètres, le nœud `invoke` possède un nœud enfant `arguments`, dont les enfants seront les valeurs de paramètres formatées à l'aide du format de valeur individuelle expliqué ci-après.

Les valeurs individuelles, notamment les paramètres de fonction et les valeurs de retour, utilisent un format qui comprend des informations sur le type de données, en plus des valeurs elles-mêmes : Le tableau suivant répertorie les classes ActionScript et le format XML utilisé pour coder des valeurs de ce type de données :

Classe/valeur ActionScript	Classe/valeur C#	Format	Commentaires
null	null	<null/>	
Boolean `true`	bool true	<true/>	
Boolean `false`	bool false	<false/>	
String	string	<string>valeur de chaîne</string>	
Number, int, uint	single, double, int, uint	`<number>27.5</number>` `<number>-12</number>`	

Array (combinaison possible de divers types d'éléments)	Une collection qui accepte des éléments de plusieurs types, par exemple ArrayList ou object[]	`<array>`
		`<property id="0">`
		`<number>27.5</number>`
		`</property>`
		`<property id="1">`
		`<string>Hello there!</string>`
		`</property>`
		`...`
		`</array>`
			Le nœud `property` définit des éléments individuels et l'attribut `id` est l'index numérique en base zéro.

Object	Un dictionnaire contenant des clés de chaîne et des valeurs d'objet, par exemple HashTable avec clés de chaînes	`<object>`
		`<property id="name">`
		`<string>John Doe</string>`
		`</property>`
		`<property id="age">`
		`<string>33</string>`
		`</property>`
		`...`
		`</object>`
			Le nœud `property` définit des propriétés individuelles et l'attribut `id` est le nom de la propriété (une chaîne).

Autres classe intégrées ou personnalisées		`<null/> or` `<object></object>`	ActionScript convertit les autres objets en valeur null ou en objet vide. Dans les deux cas, toutes les valeurs de propriété sont perdues.

Remarque : *à titre d'exemple, ce tableau indique des classes C# équivalentes en plus des classes ActionScript. Néanmoins, l'API externe peut être utilisée pour communiquer avec un langage ou un moteur d'exécution prenant en charge les contrôles ActiveX, et n'est pas limitée aux applications C#.*

Lorsque vous créez vos propres applications à l'aide de l'API externe avec une application conteneur ActiveX, il est commode d'écrire un proxy qui effectuera la tâche de conversion des appels de fonction native au format XML sérialisé. Pour consulter un exemple de classe proxy écrite dans C#, voir Fonctionnement interne de la classe ExternalInterfaceProxy.

Exemple d'API externe : communications entre ActionScript et JavaScript dans un navigateur Web

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Cette application exemple illustre les techniques de communication possibles entre ActionScript et JavaScript au sein d'un navigateur Web. Il s'agit d'une application de messagerie instantanée qui permet à l'utilisateur de discuter avec lui-même (d'où le nom de l'application : Introvert IM). L'API externe permet d'envoyer les messages entre un formulaire HTML dans la page Web et une interface SWF. Les techniques décrites dans cet exemple sont les suivantes :

- Vérification de la disponibilité du navigateur avant d'établir la communication afin de garantir une initialisation correcte
- Vérification de la prise en charge de l'API externe par le conteneur
- Appel des fonctions JavaScript à partir d'ActionScript, transmission des paramètres et réception des valeurs en retour
- Mise à disposition des méthodes ActionScript que JavaScript doit appeler et exécution de ces appels

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers de l'application Introvert IM se trouvent dans le dossier Samples/IntrovertIM_HTML. L'application se compose des fichiers suivants :

Fichier	Description
IntrovertIMApp.fla ou IntrovertIMApp.mxml	Le fichier d'application principal pour Flash (FLA) ou Flex (MXML)
com/example/programmingas3/introvertIM/IMManager.as	Classe établissant et gérant les communications entre ActionScript et le conteneur.
com/example/programmingas3/introvertIM/IMMessageEvent.as	Type d'événement personnalisé distribué par la classe IMManager à la réception d'un message du conteneur.
com/example/programmingas3/introvertIM/IMStatus.as	Enumération dont les valeurs représentent les différents statuts de disponibilité pouvant être sélectionnés dans l'application.
html-flash/IntrovertIMApp.html ou html-template/index.template.html	La page HTML pour l'application pour Flash (html-flash/IntrovertIMApp.html) ou le modèle utilisé pour créer la page HTML du conteneur pour l'application pour Adobe Flex (html-template/index.template.html). Ce fichier contient toutes les fonctions JavaScript qui constitue le conteneur de l'application.

Préparation de la communication entre ActionScript et le navigateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API externe est le plus souvent utilisée pour permettre aux applications ActionScript de communiquer avec le navigateur Web. Grâce à elle, les méthodes ActionScript peuvent appeler du code écrit dans JavaScript, et inversement. En raison de la complexité des navigateurs et de leurs processus internes de rendu des pages, il est impossible de garantir qu'un document SWF pourra enregistrer ses rappels avant l'exécution du premier code JavaScript de la page HTML. Par conséquent, avant d'appeler les fonctions du document SWF à partir de JavaScript, le document SWF doit toujours appeler la page HTML pour lui indiquer qu'il est prêt à accepter des connexions.

Par exemple, grâce à une série d'étapes effectuées par la classe IMManager, Introvert IM détermine si le navigateur est prêt à communiquer et prépare le fichier SWF à la communication. La première étape, qui vérifie que le navigateur est prêt à communiquer, a lieu dans le constructeur IMManager, comme suit :

```
public function IMManager(initialStatus:IMStatus)
{
    _status = initialStatus;

    // Check if the container is able to use the external API.
    if (ExternalInterface.available)
    {
        try
        {
            // This calls the isContainerReady() method, which in turn calls
            // the container to see if Flash Player has loaded and the container
            // is ready to receive calls from the SWF.
            var containerReady:Boolean = isContainerReady();
            if (containerReady)
            {
                // If the container is ready, register the SWF's functions.
                setupCallbacks();
            }
            else
            {
                // If the container is not ready, set up a Timer to call the
                // container at 100ms intervals. Once the container responds that
                // it's ready, the timer will be stopped.
                var readyTimer:Timer = new Timer(100);
                readyTimer.addEventListener(TimerEvent.TIMER, timerHandler);
                readyTimer.start();
            }
        }
        ...
    }
    else
    {
        trace("External interface is not available for this container.");
    }
}
```

Tout d'abord, le code vérifie si l'API externe est disponible dans le conteneur actuel à l'aide de la propriété `ExternalInterface.available`. Si c'est le cas, le processus de mise en place de la communication commence. Pour parer aux éventuelles exceptions de sécurité et autres erreurs qui peuvent se produire pendant les communications avec une application externe, le code est enveloppé dans un bloc `try` (les blocs `catch` correspondants ont été omis de l'exemple pour plus de concision).

Le code appelle ensuite la méthode `isContainerReady()`, présentée ici :

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

La méthode `isContainerReady()` utilise à son tour la méthode `ExternalInterface.call()` pour appeler la fonction JavaScript `isReady()`, comme suit :

```
<script language="JavaScript">
<!--
// ------- Private vars -------
var jsReady = false;
...
// ------- functions called by ActionScript -------
// called to check if the page has initialized and JavaScript is available
function isReady()
{
    return jsReady;
}
...
// called by the onload event of the <body> tag
function pageInit()
{
    // Record that JavaScript is ready to go.
    jsReady = true;
}
...
//-->
</script>
```

La fonction `isReady()` renvoie simplement la valeur de la variable `jsReady`. Cette variable a au départ la valeur `false`. Une fois que l'événement `onload` de la page Web est déclenché, la valeur de la variable devient `true`. En d'autres termes, si ActionScript appelle la fonction `isReady()` avant que la page soit chargée, JavaScript renvoie la valeur `false` à `ExternalInterface.call("isReady")`, à la suite de quoi la méthode ActionScript `isContainerReady()` renvoie la valeur `false`. Une fois la page chargée, la fonction JavaScript `isReady()` renvoie la valeur `true`, donc la méthode ActionScript `isContainerReady()` renvoie aussi la valeur `true`.

Dans le constructeur IMManager, deux solutions sont possibles en fonction de la disponibilité du conteneur. Si `isContainerReady()` renvoie la valeur `true`, le code appelle simplement la méthode `setupCallbacks()`, qui achève la mise en place de la communication avec JavaScript. Dans l'autre cas, si `isContainerReady()` renvoie la valeur `false`, le processus est pratiquement mis en attente. Un objet Timer est créé pour appeler la méthode `timerHandler()` toutes les 100 millisecondes, comme suit :

```
private function timerHandler(event:TimerEvent):void
{
    // Check if the container is now ready.
    var isReady:Boolean = isContainerReady();
    if (isReady)
    {
        // If the container has become ready, we don't need to check anymore,
        // so stop the timer.
        Timer(event.target).stop();
        // Set up the ActionScript methods that will be available to be
        // called by the container.
        setupCallbacks();
    }
}
```

Chaque fois que la méthode `timerHandler()` est appelée, elle vérifie à nouveau le résultat de la méthode `isContainerReady()`. Lorsque le conteneur est initialisé, la méthode renvoie la valeur `true`. Le code arrête alors le minuteur et appelle la méthode `setupCallbacks()` pour finir la mise en place de la communication avec le navigateur.

Présentation des méthodes ActionScript à JavaScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Comme le montre l'exemple précédent, une fois que le code établit que le navigateur est prêt, la méthode `setupCallbacks()` est appelée. Cette méthode prépare ActionScript pour recevoir des appels à partir de JavaScript, comme le montre cet exemple :

```
private function setupCallbacks():void
{
    // Register the SWF client functions with the container
    ExternalInterface.addCallback("newMessage", newMessage);
    ExternalInterface.addCallback("getStatus", getStatus);
    // Notify the container that the SWF is ready to be called.
    ExternalInterface.call("setSWFIsReady");
}
```

La méthode `setCallBacks()` achève la préparation de la communication avec le conteneur en appelant `ExternalInterface.addCallback()` afin d'enregistrer deux méthodes qui pourront être appelées par JavaScript. Dans ce code, le premier paramètre (le nom qui sert à désigner la méthode dans JavaScript, soit `"newMessage"` et `"getStatus"`) est identique au nom de la méthode dans ActionScript (dans ce cas, il n'y avait pas d'intérêt à utiliser des noms différents, on a donc réutilisé les mêmes noms par souci de simplification). Enfin, la méthode `ExternalInterface.call()` est utilisée pour appeler la fonction JavaScript `setSWFIsReady()`, qui avertit le conteneur que les fonctions ActionScript ont été enregistrées.

Communication d'ActionScript vers le navigateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'application Introvert IM met en évidence plusieurs exemples d'appel de fonctions JavaScript dans la page conteneur. Dans le cas le plus simple (un exemple issu de la méthode `setupCallbacks()`), la fonction JavaScript `setSWFIsReady()` est appelée sans transmettre de paramètres ni recevoir de valeur en retour :

```
ExternalInterface.call("setSWFIsReady");
```

Dans un autre exemple issu de la méthode `isContainerReady()`, ActionScript appelle la fonction `isReady()` et reçoit une valeur booléenne en réponse :

```
var result:Boolean = ExternalInterface.call("isReady");
```

Vous pouvez également transmettre des paramètres aux fonctions JavaScript à l'aide de l'API externe. Considérez par exemple la méthode `sendMessage()` de la classe IMManager, qui est appelée lorsque l'utilisateur envoie un nouveau message à son interlocuteur.

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

Là encore, `ExternalInterface.call()` sert à appeler la fonction JavaScript désignée pour avertir le navigateur du nouveau message. En outre, le message lui-même est transmis comme paramètre supplémentaire à `ExternalInterface.call()`. Il est ensuite transmis comme paramètre à la fonction JavaScript `newMessage()`.

Appel du code ActionScript à partir de JavaScript

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Une communication se fait normalement de manière bidirectionnelle, ce que respecte l'application Introvert IM. D'un côté, le client de messagerie Flash Player appelle JavaScript pour envoyer des messages, de l'autre, le formulaire HTML appelle le code JavaScript pour envoyer des messages au fichier SWF et recevoir de celui-ci des informations. Par exemple, lorsque le fichier SWF avertit le conteneur qu'il a établi le contact et peut communiquer, la première action du navigateur consiste à appeler la méthode `getStatus()` de la classe IMManager pour recevoir du client de messagerie SWF le statut de disponibilité de l'utilisateur initial. Cela se fait dans la page Web, avec la fonction `updateStatus()`, comme illustré ci-après :

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

Le code vérifie la valeur de la variable `swfReady`, qui vérifie si le fichier SWF a averti le navigateur qu'il avait enregistré ses méthodes auprès de la classe ExternalInterface. Si le fichier SWF est prêt à recevoir la communication, la ligne suivante (`var currentStatus =` ...) appelle la méthode `getStatus()` dans la classe IMManager. Trois opérations se produisent dans cette ligne de code :

- La fonction JavaScript `getSWF()` est appelée et renvoie une référence à l'objet JavaScript représentant le fichier SWF. Le paramètre transmis à `getSWF()` détermine si l'objet de navigateur est renvoyé, au cas où il y aurait plus d'un fichier SWF dans la page HTML. La valeur transmise à ce paramètre doit correspondre à l'attribut `id` de la balise `object` et à l'attribut `name` de la balise `embed`, toutes deux utilisées pour inclure le fichier SWF.
- Avec la référence au fichier SWF, la méthode `getStatus()` est appelée comme s'il s'agissait d'une méthode de l'objet SWF. Dans ce cas, le nom de fonction « `getStatus` » est utilisé, car c'est le nom sous lequel la fonction ActionScript a été enregistrée à l'aide de `ExternalInterface.addCallback()`.
- La méthode ActionScript `getStatus()` renvoie une valeur qui est attribuée à la variable `currentStatus`, laquelle devient ensuite le contenu (la valeur `value`) du champ de texte `status`.

Remarque : *si vous vous référez au code, vous avez probablement noté que dans le code source relatif à la fonction `updateStatus()`, la ligne qui appelle la fonction `getSWF()` est en fait écrite comme suit : var currentStatus = getSWF("${application}").getStatus(). Le texte `${application}` est un espace réservé dans le modèle de page HTML. Lorsqu'Adobe Flash Builder génère la page HTML en tant que telle pour l'application, cet espace réservé est remplacé par le texte faisant office d'attribut `id` de la balise `object` et d'attribut `name` de la balise `embed` (soit `IntrovertIMApp` dans l'exemple). Il s'agit de la valeur attendue par la fonction `getSWF()`.*

La fonction JavaScript `sendMessage()` illustre la transmission d'un paramètre à la fonction ActionScript (`sendMessage()` est la fonction appelée lorsque l'utilisateur appuie sur le bouton Envoyer de la page HTML).

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

La méthode ActionScript `newMessage()` attend un seul paramètre. De ce fait, la variable JavaScript `message` est transmise à ActionScript en l'utilisant comme paramètre dans l'appel de la méthode `newMessage()` du code JavaScript.

Détection du type de navigateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'accès au contenu varie d'un navigateur à l'autre. Pour cette raison, il est important de toujours utiliser JavaScript pour détecter le navigateur utilisé et accéder ensuite à la séquence selon la syntaxe propre au navigateur à l'aide de l'objet de fenêtre ou de document, comme le montre la fonction JavaScript `getSWF()` de cet exemple :

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

Si votre script ne détecte pas le type de navigateur de l'utilisateur, ce dernier peut noter un comportement inattendu lors de la lecture des fichiers SWF dans un conteneur HTML.

Chapitre 48 : Validation des signatures XML dans AIR

Adobe AIR 1.5 et les versions ultérieures

Les classes de l'API XMLSignatureValidator d'Adobe® AIR® permettent de valider les signatures numériques conformes à un sous-ensemble de la recommandation du W3C portant sur la syntaxe et le traitement des signatures XML (http://www.w3.org/TR/xmldsig-core/). Des signatures XML peuvent être utilisées pour vérifier l'intégrité et l'identité du signataire des données ou des informations.

Les signatures XML peuvent être utilisées pour valider les messages ou les ressources téléchargé(e)s par votre application. Par exemple, si votre application fournit des services sur la base d'un abonnement, vous pouvez encapsuler les termes de l'abonnement dans un document XML signé. Si quelqu'un tente de modifier le document d'abonnement, la validation échoue.

Vous pouvez également utiliser une signature XML pour simplifier la validation des ressources téléchargées par votre application en incluant un fichier manifest signé contenant les digests de ces ressources. Votre application peut alors vérifier que les ressources n'ont pas été modifiées en comparant le digest du fichier signé et le digest calculé à partir des octets chargés. Cette technique se révèle particulièrement utile lorsque la ressource téléchargée est un fichier SWF ou un autre contenu actif à exécuter dans le sandbox de sécurité de l'application.

Bases de la validation des signatures XML

Adobe AIR 1.5 et les versions ultérieures

Pour obtenir une explication rapide de la validation des signatures XML, ainsi que des exemples de code correspondants, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Création et validation de signatures XML (Flex)
- Création et validation de signatures XML (Flash) (disponible en anglais uniquement)

Adobe® AIR® fournit la classe XMLSignatureValidator et l'interface IURIDereferencer pour la validation des signatures XML. La syntaxe XML acceptée par la classe XMLSignatureValidator est un sous-ensemble de la recommandation du W3C portant sur la syntaxe et le traitement des signatures XML. (Comme seul un sous-ensemble de la recommandation est pris en charge, toutes les signatures valides ne peuvent pas être validées.) AIR ne fournit pas d'API capable de créer des signatures XML.

Classes de validation des signatures XML

Adobe AIR 1.5 et les versions ultérieures

L'API de validation des signatures XML comprend les classes suivantes :

Package	Classes
flash.security	• XMLSignatureValidator • IURIDereferencer (interface) Les constantes de chaîne XMLSignatureValidator sont définies dans les classes suivantes : • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Event • ErrorEvent

Utilisation des classes de validation des signatures XML

Adobe AIR 1.5 et les versions ultérieures

Pour utiliser la classe XMLSignatureValidator pour valider une signature XML, vous devez :

- Créer un objet XMLSignatureValidator
- Fournir une implémentation de l'interface IURIDereferencer. L'objet XMLSignatureValidator appelle la méthode IURIDereferencer `dereference()`, en transmettant l'URI pour chaque référence présente dans une signature. La méthode `dereference()` doit résoudre l'URI et renvoyer les données référencées (situées dans le même document que la signature ou dans une ressource externe).
- Définir les paramètres de certificat de confiance, de vérification de la révocation et de validation des références de l'objet XMLSignatureValidator selon les besoins de votre application.
- Ajouter des écouteurs d'événement pour les événements `complete` et `error`.
- Appeler la méthode `verify()`, en transmettant la signature à vérifier.
- Traiter les événements `complete` et `error` et interpréter les résultats.

L'exemple suivant implémente une fonction `validate()` qui vérifie la validité d'une signature XML. Les propriétés XMLSignatureValidator sont définies de telle sorte que le certificat de signature doit être présent dans le magasin d'approbation du système, ou chaîné vers un certificat du magasin d'approbation. L'exemple suppose également l'existence d'une classe IURIDereferencer appropriée nommée *XMLDereferencer*.

```
private function validate( xmlSignature:XML ):void
{
    var verifier:XMLSignatureValidator = new XMLSignatureValidator();
    verifier.addEventListener(Event.COMPLETE, verificationComplete);
    verifier.addEventListener(ErrorEvent.ERROR, verificationError);
    try
    {
        verifier.uriDereferencer = new XMLDereferencer();

        verifier.referencesValidationSetting =
            ReferencesValidationSetting.VALID_IDENTITY;
        verifier.revocationCheckSetting = RevocationCheckSettings.BEST_EFFORT;
        verifier.useSystemTrustStore = true;

        //Verify the signature
        verifier.verify( xmlSignature );
    }
    catch (e:Error)
        {
            trace("Verification error.\n" + e);
        }
}

//Trace verification results
private function verificationComplete(event:Event):void

    var signature:XMLSignatureValidator = event.target as XMLSignatureValidator;
    trace("Signature status: " + signature.validityStatus + "\n");
    trace("  Digest status: " + signature.digestStatus + "\n");
    trace("  Identity status: " + signature.identityStatus + "\n");
    trace("  Reference status: " + signature.referencesStatus + "\n");
}

private function verificationError(event:ErrorEvent):void
{
    trace("Verification error.\n" + event.text);
}
```

Processus de validation des signatures XML

Adobe AIR 1.5 et les versions ultérieures

Lorsque vous appelez la méthode `verify()` de la classe XMLSignatureValidator, AIR procède comme suit :

- Le moteur d'exécution vérifie l'intégrité cryptographique de la signature à l'aide de la clé publique du certificat.
- Le moteur d'exécution établie l'intégrité cryptographique, l'identité et la véracité du certificat sur la base des paramètres actuels de l'objet XMLSignatureValidator.

  La confiance accordée au certificat de signature est essentielle pour l'intégrité du processus de validation. La validation de la signature est effectuée via un processus cryptographique bien défini, mais la fiabilité du certificat de signature ne peut pas être assurée par un algorithme.

  En général, trois méthodes permettent de savoir si un certificat est fiable :

  - S'appuyer sur des autorités de certification et sur le magasin d'approbations du système d'exploitation.

- Obtenir directement du signataire une copie du certificat, un autre certificat jouant le rôle d'ancre de confiance pour le certificat ou des informations permettant d'identifier le certificat avec certitude, par exemple la clé publique.
- Demander à l'utilisateur final de votre application s'il fait confiance au certificat. Une telle requête n'est pas valide pour les certificats auto-signés car les informations d'identification du certificat ne sont pas fiables par nature.

- Le moteur d'exécution vérifie l'intégrité cryptographique des données signées.

  Les données signées sont vérifiées à l'aide de votre implémentation d'IURIDereferencer. Pour chaque référence du document de signature, la méthode `dereference()` de l'implémentation d'IURIDereferencer est appelée. Les données renvoyées par la méthode `dereference()` sont utilisées pour calculer le digest de référence. La valeur du digest est comparée au digest enregistré dans le document de signature. Si les digests correspondent, les données n'ont pas été modifiées depuis leur signature.

  Une considération importante lorsque l'on s'appuie sur les résultats de la validation d'une signature XML est que seul ce qui a été signé est sécurisé. Par exemple, prenons le cas de la liste signée des fichiers contenus dans un package. Lorsque XMLSignatureValidator vérifie la signature, l'opération s'assure uniquement que la liste elle-même n'a pas été modifiée. Les données des fichiers n'étant pas signées, la signature reste valide même si les fichiers référencés ont été modifiés ou supprimés.

  Remarque : *pour vérifier les fichiers d'une telle liste, vous pouvez calculer le digest de leurs données (en utilisant le même algorithme de hachage que pour la liste) et comparer le résultat au digest stocké dans la liste signée. Dans certains cas, vous pouvez également vérifier la présence de fichiers supplémentaires.*

Interprétation des résultats de la validation

Adobe AIR 1.5 et les versions ultérieures

Les résultats de la validation sont indiqués par les propriétés d'état de l'objet XMLSignatureValidator. Ces propriétés peuvent être lues après la distribution de l'événement *complete* par l'objet validateur. Les quatre propriétés d'état sont : `validityStatus`, `digestStatus`, `identityStatus` et `referencesStatus`.

Propriété validityStatus

Adobe AIR 1.5 et les versions ultérieures

La propriété `validityStatus` renvoie la validité générale de la signature. Cette propriété `validityStatus` dépend de l'état des trois autres propriétés d'état et peut prendre l'une des valeurs suivantes :

- `valid` : si les propriétés `digestStatus`, `identityStatus` et `referencesStatus` sont toutes `valid`.
- `invalid` : si l'une des propriétés d'état est `invalid`.
- `unknown` : si l'une des propriétés d'état est `unknown` et qu'aucun état individuel n'est `invalid`.

Propriété digestStatus

Adobe AIR 1.5 et les versions ultérieures

La propriété `digestStatus` renvoie le résultat de la vérification cryptographique du digest du message. Cette propriété `digestStatus` peut prendre l'une des valeurs suivantes :

- `valid` : si le document de signature lui-même n'a pas été modifié depuis la signature.
- `invalid` : si le document de signature a été modifié ou est incorrect.

- `unknown` : si la méthode `verify()` ne s'est pas terminée sans erreur.

Propriété identityStatus

Adobe AIR 1.5 et les versions ultérieures

La propriété `identityStatus` renvoie l'état du certificat de signature. La valeur de cette propriété dépend de plusieurs facteurs :

- intégrité cryptographique du certificat ;
- expiration ou révocation éventuelle du certificat ;
- approbation du certificat sur l'ordinateur en cours ;
- état de l'objet XMLSignatureValidator (par exemple, si des certificats supplémentaires ont été ajoutés pour définir la chaîne de confiance, si ces certificats sont approuvés, et valeurs des propriétés `useSystemTrustStore` et `revocationCheckSettings`).

La propriété `identityStatus` peut avoir l'une des valeurs suivantes :

- `valid` : pour être considéré comme valide, le certificat de signature doit répondre aux conditions suivantes :
  - Le certificat de signature ne doit pas avoir été modifié.
  - Le certificat de signature ne doit pas être arrivé à expiration ou avoir été révoqué, sauf si un horodatage valide est présent dans la signature. Dans ce dernier cas, le certificat est considéré comme valide pour autant qu'il l'ait été au moment de la signature du document. (Le certificat utilisé par le service d'horodatage pour signer l'horodatage doit être lié par une chaîne de certificats à un certificat racine de confiance sur l'ordinateur de l'utilisateur.)
  - Le certificat de signature est approuvé. Un certificat est approuvé lorsqu'il est situé dans le magasin d'approbation du système ou qu'il chaîne vers un autre certificat du magasin d'approbation et que la propriété `useSystemTrustStore` est définie sur true. Vous pouvez également désigner un certificat comme étant approuvé en utilisant la méthode `addCertificate()` de l'objet XMLSignatureValidator.
  - Le certificat est, en fait, le certificat de signature.
- `invalid` : le certificat est arrivé à expiration ou a été révoqué (et aucun horodatage n'assure sa validité au moment de la signature) ou le certificat a été modifié.
- `unknown` : si le certificat n'est pas invalide, mais qu'il n'est pas non plus approuvé. Les certificats auto-signés, par exemple, sont signalés comme `unknown` (sauf s'ils sont explicitement approuvés). La propriété `identityStatus` est également désignée comme `unknown` si la méthode `verify()` ne s'est pas terminée sans erreur ou si l'identité n'a pas été vérifiée parce que le digest de la signature n'est pas valide.

Propriété referencesStatus

Adobe AIR 1.5 et les versions ultérieures

La propriété `referencesStatus` renvoie l'intégrité cryptographique des références présentes dans l'élément SignedData de la signature.

- `valid` : si le digest calculé de chaque référence de la signature correspond au digest enregistré correspondant dans la signature XML. Un état `valid` indique que les données signées n'ont pas été modifiées.
- `invalid` : si l'un des digests calculés ne correspond pas au digest correspondant dans la signature.

- `unknown` : si les digests des références n'ont pas été vérifiés. Les références ne sont pas vérifiées si le digest général de la signature est `invalid` ou si le certificat de signature n'est pas valide. Si la propriété `identityStatus` est `unknown`, les références ne sont vérifiées que si la valeur de `referencesValidationSetting` est `validOrUnknown`.

A propos des signatures XML

Adobe AIR 1.5 et les versions ultérieures

Une signature XML est une signature numérique représentée en syntaxe XML. Les données d'une signature XML peuvent être utilisées pour garantir la non modification des informations signées depuis la signature. De plus, lorsqu'un certificat de signature a été publié par une autorité de certification approuvée, l'identité du signataire peut être vérifiée via l'infrastructure de clé publique.

Une signature XML peut être appliquée à n'importe quel type de données numériques (au format binaire ou XML). Des signatures XML sont généralement utilisées pour :

- vérifier si des ressources externes ou téléchargées ont été modifiées ;
- vérifier que des messages proviennent d'une source connue ;
- valider les droits de licence d'une application ou les droits d'abonnement.

Syntaxe de signature XML prise en charge

Adobe AIR 1.5 et les versions ultérieures

AIR prend en charge les éléments suivants de la recommandation W3C portant sur la syntaxe et le traitement des signatures :

- Tous les éléments de syntaxe de signature centraux (section 4 du document de la recommandation W3C uniquement), à l'exception de l'élément KeyInfo qui n'est pas entièrement pris en charge
- L'élément KeyInfo ne doit contenir qu'un élément X509Data.
- Un élément X509Data ne doit contenir qu'un élément X509Certificate.
- La méthode digest SHA256
- L'algorithme de signature RSA-SHA1 (PKCS1)
- La méthode de canonisation « XML canonique sans commentaires » et l'algorithme de transformation
- La transformation de la signature enveloppée
- horodatage

Le document suivant présente une signature XML typique (la plupart des données cryptographiques ont été supprimées pour simplifier l'exemple) :

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <Reference URI="URI_to_signed_data">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/></Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>uoo...vY=</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>Ked...w==</SignatureValue>
    <KeyInfo>
        <X509Data>
            <X509Certificate>i7d...w==</X509Certificate>
        </X509Data>
    </KeyInfo>
</Signature>
```

Les éléments-clés d'une signature sont les suivants :

- SignedInfo : contient les références aux données signées et les valeurs du digest calculé au moment de la signature. Les données signées elles-mêmes peuvent être incluses dans le même document que la signature XML ou peuvent être externes.
- SignatureValue : contient un digest de l'élément SignedInfo chiffré avec la clé privée du signataire.
- KeyInfo : contient le certificat de signature et tout certificat supplémentaire nécessaire pour établir la chaîne d'approbation. Notez que, bien que l'élément KeyInfo soit techniquement facultatif, AIR ne peut pas valider la signature s'il n'est pas inclus.

Il existe trois types généraux de signature XML :

- Enveloppée : la signature est insérée dans les données XML signées.
- Enveloppante : les données XML signées sont incluses dans un élément Object au sein de l'élément Signature.
- Détachée : les données signées sont externes à la signature XML. Les données signées doivent être dans un fichier externe. Elles peuvent également être dans le même document XML que la signature, mais pas dans un élément parent ou enfant de l'élément Signature.

Pour référencer les données signées, les signatures XML utilisent des URI. Pour résoudre ces URI, les applications de signature et de validation doivent utiliser les mêmes conventions. Lorsque vous utilisez la classe XMLSignatureValidator, vous devez fournir une implémentation de l'interface IURIDereferencer. Cette implémentation est chargée de résoudre l'URI et de renvoyer les données signées sous forme d'objet ByteArray. Le digest de l'objet ByteArray renvoyé est calculé avec le même algorithme que celui qui a permis de calculer le digest de la signature.

Certificats et approbation

Adobe AIR 1.5 et les versions ultérieures

Un certificat est constitué d'une clé publique, d'informations d'identification et, éventuellement, d'un ou plusieurs certificats appartenant à l'autorité de certification émettrice.

Deux méthodes permettent d'établir la confiance d'un certificat. Vous pouvez établir la confiance en obtenant directement une copie du certificat auprès du signataire, par exemple sur un support physique, ou par l'intermédiaire d'une transmission numérique sécurisée, telle qu'une transaction SSL. Vous pouvez également vous appuyer sur une autorité de certification pour déterminer la fiabilité du certificat de signature.

Dans ce cas, le certificat de signature doit être publié par une autorité approuvée sur l'ordinateur sur lequel la signature est validée. La plupart des éditeurs de systèmes d'exploitation placent les certificats racine d'un certain nombre d'autorités de certification dans le magasin d'approbation du système. Les utilisateurs peuvent également ajouter et supprimer des certificats dans ce magasin.

Même si le certificat est publié par une autorité de certification approuvée, vous devez décider si le certificat appartient à une personne de confiance. Dans la plupart des cas, cette décision revient à l'utilisateur final. Par exemple, lors de l'installation d'une application AIR, le programme d'installation AIR affiche les informations d'identification du certificat de l'éditeur lorsqu'il invite l'utilisateur à indiquer s'il souhaite installer l'application. Dans d'autres cas, vous devrez peut-être comparer la clé publique ou d'autres informations de certificat à une liste de clés acceptables. (Cette liste doit être sécurisée, éventuellement par sa propre signature ou en étant stockée dans le magasin local chiffré d'AIR, de sorte que la liste elle-même ne puisse pas être altérée.)

Remarque : *bien que vous puissiez choisir d'approuver le certificat de signature sans vérification indépendante (par exemple lorsque une signature est « auto-signée »), la vérification de la signature n'est pas suffisamment rassurante dans ce cas. Si vous ne connaissez pas l'auteur de la signature, la certitude que la signature n'a pas été modifiée n'a que peu de valeur. La signature peut être une contrefaçon signée.*

Expiration et révocation des certificats

Adobe AIR 1.5 et les versions ultérieures

Tous les certificats arrivent à expiration à un moment donné. Les certificats peuvent également être révoqués par l'autorité de certification émettrice si, par exemple, la clé privée associée au certificat a été compromise ou volée. Si une signature est signée avec un certificat arrivé à expiration ou révoqué, elle est désignée comme non valide sauf si un horodatage a été inclus dans la signature. En présence d'un horodatage, la classe XMLSignatureValidator valide la signature si le certificat était valide au moment de la signature.

Un horodatage est un message numérique signé provenant d'un service d'horodatage qui certifie que les données ont été signées à une heure et une date spécifiques. Les horodatages sont publiés par des autorités d'horodatage et signés par le propre certificat de ces autorités. Le certificat de l'autorité d'horodatage intégré à l'horodatage doit être approuvé sur l'ordinateur en cours pour que cet horodatage soit considéré comme valide. L'objet XMLSignatureValidator ne fournit pas d'API pour désigner un autre certificat à utiliser pour valider l'horodatage.

Implémentation de l'interface IURIDereferencer

Adobe AIR 1.5 et les versions ultérieures

Pour valider une signature XML, vous devez fournir une implémentation de l'interface IURIDereferencer. Cette implémentation est chargée de résoudre les URI spécifiées dans les éléments Reference d'un document de signature XML, puis de renvoyer les données pour que le digest puisse être calculé. Le digest calculé est comparé au digest de la signature pour déterminer si les données référencées ont été modifiées depuis la création de la signature.

Remarque : *les applications AIR de type HTML doivent importer une bibliothèque SWF contenant une implémentation ActionScript pour valider les signatures XML. L'interface IURIDereferencer ne peut pas être implémentée en JavaScript.*

L'interface IURIDerefencer possède une méthode unique, `dereference(uri:String)`, qui doit être implémentée. L'objet XMLSignatureValidator appelle cette méthode pour chaque référence présente dans la signature. La méthode doit renvoyer les données dans un objet ByteArray.

Dans la plupart des cas, vous devrez également ajouter des propriétés ou des méthodes permettant à votre objet déréférenceur de localiser les données référencées. Par exemple, si les données signées sont dans le même document que la signature, vous pouvez ajouter une variable de membre fournissant une référence au document XML. La méthode `dereference()` peut alors utiliser cette variable, ainsi que l'URI, pour localiser les données référencées. De la même façon, si les données signées sont situées dans un répertoire du système de fichiers local, la méthode `dereference()` peut avoir besoin d'une propriété fournissant le chemin conduisant à ce répertoire pour résoudre les fichiers référencés.

L'objet XMLSignatureValidator repose entièrement sur l'objet déréférenceur pour interpréter les chaînes URI. Les règles standard du déréférencement des URI sont données dans la section 4.3.3 de la recommandation du W3C portant sur la syntaxe et le traitement des signatures XML.

Déréférencement des URI dans les signatures enveloppées

Adobe AIR 1.5 et les versions ultérieures

Lorsqu'une signature XML enveloppée est générée, ses éléments sont insérés dans les données signées. Par exemple, si vous signez le message suivant à l'aide d'une structure de signature enveloppée :

```
<message>
    <data>...</data>
</message>
```

Le document signé résultant ressemblera à cela :

```
<message>
    <data>...</data>
     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="">
                <Transforms>
                    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <DigestValue>yv6...Z0Y=</DigestValue>
            </Reference>
        </SignedInfo>
        <SignatureValue>cCY...LQ==</SignatureValue>
        <KeyInfo>
            <X509Data>
                <X509Certificate>MII...4e</X509Certificate>
            </X509Data>
        </KeyInfo>
    </Signature>
</message>
```

Remarquez que la signature contient un élément Reference avec une chaîne vide pour son URI. Dans ce contexte, une chaîne vide fait référence à la racine du document.

Notez également que l'algorithme de transformation spécifie qu'une transformation de signature enveloppée a été appliquée. Lorsqu'une telle transformation a été appliquée, l'objet XMLSignatureValidator supprime automatiquement la signature du document avant de calculer le digest. Cela signifie que le déréférenceur n'a pas besoin de supprimer l'élément Signature lorsqu'il renvoie les données.

L'exemple suivant présente un déréférenceur pour signatures enveloppées :

```
package
{
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class EnvelopedDereferencer
        extends EventDispatcher implements IURIDereferencer
    {
        private var signedMessage:XML;

        public function EnvelopedDereferencer( signedMessage:XML )
        {
            this.signedMessage = signedMessage;
        }

        public function dereference( uri:String ):IDataInput
        {
            try
            {
                if( uri.length != 0 )
                {
                    throw( new Error("Unsupported signature type.") );
                }
                var data:ByteArray = new ByteArray();
                data.writeUTFBytes( signedMessage.toXMLString() );
                data.position = 0;
            }
            catch (e:Error)
                {
                var error:ErrorEvent =
                    new ErrorEvent("Ref error " + uri + " ", false, false, e.message);
                this.dispatchEvent(error);
                data = null;
                throw new Error("Reference not resolvable: " + uri + ", " + e.message);
            }
            finally
            {
                return data;
            }
        }
    }
}
```

Cette classe de déréférenceur utilise une fonction constructeur avec un paramètre, `signedMessage`, pour mettre le document de signature enveloppée à la disposition de la méthode `dereference()`. Comme la référence placée dans une signature enveloppée fait toujours référence à la racine des données signées, la méthode `dereferencer()` écrit le document dans un tableau d'octets et renvoie celui-ci.

Déréférencement des URI dans des signatures enveloppantes et détachées

Adobe AIR 1.5 et les versions ultérieures

Lorsque les données signées sont dans le même document que la signature elle-même, les URI des références utilisent généralement la syntaxe XPath ou XPointer pour traiter les éléments signés. La recommandation du W3C portant sur la syntaxe et le traitement des signatures XML ne conseillant que cette syntaxe, il est préférable de baser votre implémentation sur les signatures que vous envisagez de rencontrer (et d'ajouter suffisamment de vérification d'erreur pour traiter intelligemment la syntaxe non prise en charge).

La signature d'une application AIR est un exemple de signature enveloppante. Les fichiers de l'application sont répertoriés dans un élément Manifest. L'élément Manifest est traité dans l'attribut Reference URI à l'aide de la chaîne, « #PackageContents », qui fait référence à l'identifiant de l'élément Manifest :

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="PackageSignature">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
        <SignatureMethod Algorithm="http://www.w3.org/TR/xmldsig-core#rsa-sha1"/>
        <Reference URI="#PackageContents">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>ZMGqQdaRKQc1HirIRsDpeBDlaElS+pPotdziIAyAYDk=</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue Id="PackageSignatureValue">cQK...7Zg==</SignatureValue>
    <KeyInfo>
        <X509Data>
            <X509Certificate>MII...T4e</X509Certificate>
        </X509Data>
    </KeyInfo>
    <Object>
    <Manifest Id="PackageContents">
        <Reference URI="mimetype">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>0/oCb84THKMagtI0Dy0KogEu92TegdesqRr/clXct1c=</DigestValue>
        </Reference>
        <Reference URI="META-INF/AIR/application.xml">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>P9MqtqSdqcqnFgeoHCJysLQu4PmbUW2JdAnc1WLq8h4=</DigestValue>
        </Reference>
        <Reference URI="XMLSignatureValidation.swf">
            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">
            </DigestMethod>
            <DigestValue>OliRHRAgc9qt3Dk0m0Bi53Ur5ur3fAweIFwju74rFgE=</DigestValue>
        </Reference>
    </Manifest>
</Object>
</Signature>
```

Pour valider cette signature, un déréférenceur doit prendre la chaîne URI conteneur, « `#PackageContents` » provenant de l'élément Reference, puis renvoyer l'élément Manifest dans un objet ByteArray. Le symbole « # » fait référence à la valeur d'un attribut Id d'élément.

L'exemple suivant implémente un déréférenceur pour valider des signatures d'application AIR. Pour rester simple, l'implémentation s'appuie sur la structure connue d'une signature AIR. Un déréférenceur d'objectif général pourrait être beaucoup plus complexe.

```
package
{
    import flash.events.ErrorEvent;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;

    public class AIRSignatureDereferencer implements IURIDereferencer {
        private const XML_SIG_NS:Namespace =
            new Namespace( "http://www.w3.org/2000/09/xmldsig#" );
        private var airSignature:XML;

        public function AIRSignatureDereferencer( airSignature:XML ) {
            this.airSignature = airSignature;
        }

        public function dereference( uri:String ):IDataInput {
            var data:ByteArray = null;
            try
            {
                if( uri != "#PackageContents" )
                {
                    throw( new Error("Unsupported signature type.") );
                }
                var manifest:XMLList =
                    airSignature.XML_SIG_NS::Object.XML_SIG_NS::Manifest;
                data = new ByteArray();
                data.writeUTFBytes( manifest.toXMLString());
                data.position = 0;
            }
            catch (e:Error)
            {
                data = null;
                throw new Error("Reference not resolvable: " + uri + ", " + e.message);
            }
            finally
            {
                return data;
            }

        }
    }
}
```

Lorsque vous vérifiez ce type de signature, seules les données de l'élément Manifest sont validées. Les fichiers stockés dans le package ne sont absolument pas vérifiés. Pour vérifier l'altération éventuelle des fichiers du package, vous devez lire ces fichiers, calculer leur digest SHA256 et comparer le résultat au digest enregistré dans l'élément Manifest. L'objet XMLSignatureValidator ne vérifie pas automatiquement de telles références secondaires.

Remarque : *cet exemple n'est fourni que pour illustrer le processus de validation d'une signature. Il n'a que peu d'utilité dans une application AIR validant sa propre signature. Si l'application a déjà été modifiée, l'agent de modification peut simplement supprimer la vérification de la validation.*

Calcul des valeurs digest pour des ressources externes

Adobe AIR 1.5 et les versions ultérieures

AIR ne comprend pas de fonction intégrée pour le calcul des digests SHA256, mais le kit SDK Flex comprend une classe d'utilitaires SHA256. Le kit SDK comprend également la classe d'utilitaires encodeurs Base64 très utile pour comparer le digest calculé au digest stocké dans une signature.

L'exemple de fonction suivant lit et valide les fichiers d'un manifeste de package AIR :

```
import mx.utils.Base64Encoder;
import mx.utils.SHA256;

private function verifyManifest( sigFile:File, manifest:XML ):Boolean
{
    var result:Boolean = true;
    var message:String = '';
    var nameSpace:Namespace = manifest.namespace();

    if( manifest.nameSpace::Reference.length() <= 0 )
    {
        result = false;
        message = "Nothing to validate.";
    }
    for each (var reference:XML in manifest.nameSpace::Reference)
    {
        var file:File = sigFile.parent.parent.resolvePath( reference.@URI );
        var stream:FileStream = new FileStream();
        stream.open(file, FileMode.READ);
        var fileData:ByteArray = new ByteArray();
        stream.readBytes( fileData, 0, stream.bytesAvailable );

        var digestHex:String = SHA256.computeDigest( fileData );
        //Convert hexidecimal string to byte array
        var digest:ByteArray = new ByteArray();
        for( var c:int = 0; c < digestHex.length; c += 2 ){
            var byteChar:String = digestHex.charAt(c) + digestHex.charAt(c+1);
            digest.writeByte( parseInt( byteChar, 16 ));
        }
        digest.position = 0;
```

```
        var base64Encoder:Base64Encoder = new Base64Encoder();
        base64Encoder.insertNewLines = false;
        base64Encoder.encodeBytes( digest, 0, digest.bytesAvailable );
        var digestBase64:String = base64Encoder.toString();
        if( digestBase64 == reference.nameSpace::DigestValue )
        {
            result = result && true;
            message += "   " + reference.@URI + " verified.\n";
        }
        else
        {
            result = false;
            message += " ---- " + reference.@URI + " has been modified!\n";
        }
        base64Encoder.reset();
    }
    trace( message );
    return result;
}
```

La fonction exécute une boucle sur toutes les références de l'élément Manifest. Pour chaque référence, le digest SHA256 est calculé, encodé au format base64, puis comparé au digest du manifeste. Les URI d'un package AIR font référence aux chemins relatifs vers le répertoire de l'application. Les chemins sont résolus en fonction de l'emplacement du fichier de signature, toujours situé dans le sous-répertoire META-INF du répertoire de l'application. Notez que la classe Flex SHA256 renvoie le digest sous la forme d'une chaîne de caractères hexadécimale. Cette chaîne doit être convertie en ByteArray contenant les octets représentés par la chaîne hexadécimale.

Pour utiliser les classes mx.utils.SHA256 et Base64Encoder dans Flash CS4, vous pouvez localiser et copier ces classes dans le répertoire de développement de votre application ou compiler une bibliothèque SWF contenant les classes à l'aide du kit SDK Flex.

Déréférencement des URI dans des signatures détachées référençant des données externes

Adobe AIR 1.5 et les versions ultérieures

Lorsqu'une URI fait référence à une ressource externe, les données doivent être accédées et chargées dans un objet ByteArray. Si l'URI contient une URL absolue, il s'agit simplement de lire un fichier ou de demander une URL. Si, comme dans la plupart des cas, l'URI contient un chemin relatif, votre implémentation IURIDereferencer doit permettre de résoudre les chemins conduisant aux fichiers signés.

L'exemple suivant utilise un objet File initialisé lorsque l'occurrence de dereferencer est construite en tant que base de résolution des fichiers signés.

```
package
{
    import flash.events.ErrorEvent;
    import flash.events.EventDispatcher;
    import flash.filesystem.File;
    import flash.filesystem.FileMode;
    import flash.filesystem.FileStream;
    import flash.security.IURIDereferencer;
    import flash.utils.ByteArray;
    import flash.utils.IDataInput;
    public class RelativeFileDereferencer
        extends EventDispatcher implements IURIDereferencer
    {
        private var base:File;

        public function RelativeFileDereferencer( base:File )
        {
            this.base = base;
        }

        public function dereference( uri:String ):IDataInput
        {
            var data:ByteArray = null;
            try{
                var referent:File = this.base.resolvePath( uri );
                var refStream:FileStream = new FileStream();
                data = new ByteArray();
                refStream.open( referent, FileMode.READ );

                refStream.readBytes( data, 0, data.bytesAvailable );

            } catch ( e:Error ) {
                data = null;
                throw new Error("Reference not resolvable: " + referent.nativePath + ", " +
e.message );
            } finally {
                return data;
            }
        }
    }
}
```

La fonction `dereference()` localise simplement le fichier visé par l'URI de référence, charge le contenu du fichier dans un tableau d'octets, puis renvoie cet objet ByteArray.

Remarque : *avant de valider des références externes à distance, voyez si votre application peut être vulnérable à une attaque de téléphone personnel ou du même type par un document de signature créé à des fins malveillantes.*

Chapitre 49 : Environnement du système client

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Ce chapitre explique comment interagir avec le système utilisateur. Il vous montre comment déterminer les fonctions prises en charge et comment construire des applications multilingues à l'aide de l'éditeur de la méthode d'entrée installée (IME) de l'utilisateur (le cas échéant). Il vous présente enfin les utilisations typiques des domaines d'application.

Voir aussi

flash.system.System

flash.system.Capabilities

Principes de base de l'environnement du système client

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Au fur et à mesure que vous créez des applications plus avancées, il se peut que vous souhaitiez en savoir plus sur les systèmes d'exploitation de vos utilisateurs et leurs fonctions d'accès. Le package flash.system contient un ensemble de classes permettant d'accéder à des fonctions de niveau système telles que :

- Détermination de l'application et du domaine de sécurité dans lesquels le code est exécuté
- Détermination des fonctionnalités de l'occurrence du moteur d'exécution Flash (tel que Flash® Player ou Adobe® AIR™) de l'utilisateur (taille de l'écran - résolution) et des fonctionnalités disponibles (audio MP3, par exemple)
- Création de sites multilingues utilisant l'IME
- Interaction avec le conteneur du moteur d'exécution Flash (qui peut être une page HTML ou une application de conteneur)
- Enregistrement d'informations dans le Presse-papiers de l'utilisateur

Le package flash.system comprend aussi les classes IMEConversionMode et SecurityPanel. Ces classes contiennent des constantes statiques que vous pouvez utiliser avec les classes IME et de sécurité, respectivement.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants :

Système d'exploitation Programme principal qui est exécuté sur un ordinateur, dans lequel toutes les autres applications sont exécutées (Microsoft Windows, Mac OS X ou Linux®, par exemple).

Presse-papiers Conteneur du système d'exploitation qui contient du texte ou des éléments qui sont copiés ou coupés, et à partir duquel des éléments sont collés dans des applications.

Domaine d'application Mécanisme permettant de séparer les classes utilisées dans différents fichiers SWF de façon à ce que si les fichiers SWF comprennent différentes classes portant le même nom, elles ne se remplacent pas mutuellement.

IME (Input Method Editor) Programme (ou outil de système d'exploitation) utilisé pour entrer des caractères ou des symboles complexes par le biais d'un clavier standard.

Système client En termes de programmation, un client est la partie d'une application (ou l'application entière) exécutée sur un ordinateur et utilisée par un seul utilisateur. Le système client est le système d'exploitation sous-jacent sur l'ordinateur de l'utilisateur.

Utilisation de la classe System

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe System contient des méthodes et propriétés qui permettent d'interagir avec le système d'exploitation de l'utilisateur et de récupérer l'utilisation mémoire actuelle du moteur d'exécution. Les méthodes et propriétés de la classe System permettent aussi d'écouter les événements `imeComposition`, d'indiquer au moteur d'exécution de charger des fichiers texte externes à l'aide de la page de code active de l'utilisateur ou d'Unicode, ou de définir le contenu du Presse-papiers de l'utilisateur.

Obtention de données sur le système de l'utilisateur pendant l'exécution

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En vérifiant la propriété `System.totalMemory`, vous pouvez déterminer la quantité de mémoire (en octets) que le moteur d'exécution utilise actuellement. Cette propriété vous permet de surveiller l'utilisation mémoire et d'optimiser vos applications en fonction de ses variations. Par exemple, si un effet visuel particulier utilise une importante quantité de mémoire, vous pouvez envisager de le modifier ou de le supprimer entièrement.

La propriété `System.ime` est une référence à l'IME actuellement installé. Elle vous permet d'écouter les événements `imeComposition` (`flash.events.IMEEvent.IME_COMPOSITION`) à l'aide de la méthode `addEventListener()`.

La troisième propriété dans la classe System est `useCodePage`. Si `useCodePage` est défini sur `true`, le moteur d'exécution utilise la page de code classique du système d'exploitation pour charger les fichiers texte externes. Si vous lui attribuez la valeur `false`, vous indiquez au moteur d'exécution d'interpréter le fichier externe au format Unicode.

Si vous définissez `System.useCodePage` sur `true`, souvenez-vous que la page de code classique du système d'exploitation doit inclure les caractères utilisés dans votre fichier texte externe afin d'afficher le texte. Par exemple, si vous chargez un fichier texte externe contenant des caractères chinois, ceux-ci ne peuvent s'afficher sur un système qui utilise la page de code anglaise de Windows car elle ne comprend pas les caractères chinois.

Pour que les utilisateurs puissent afficher les fichiers texte externes utilisés dans l'application, quelle que soit la plate-forme, vous devez coder tous les fichiers texte externes en Unicode et conserver la valeur par défaut `false` de la propriété `System.useCodePage`. Le moteur d'exécution interprète ainsi le texte au format Unicode.

Enregistrement du texte dans le Presse-papiers

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe System inclut une méthode appelée `setClipboard()` qui permet au moteur d'exécution Flash de placer une chaîne spécifique dans le Presse-papiers de l'utilisateur. Pour des raisons de sécurité, il n'existe pas de méthode `Security.getClipboard()` car elle donnerait la possibilité d'accéder aux dernières données copiées dans le Presse-papiers de l'utilisateur.

Le code suivant illustre comment un message d'erreur peut être copié dans le Presse-papiers de l'utilisateur lorsqu'une erreur de sécurité survient. Le message d'erreur permettra à l'utilisateur de signaler un bogue potentiel dans une application.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 et AIR 1.0
La classe Clipboard permet de lire et d'écrire les données du Presse-papiers en réponse à un événement utilisateur. Dans AIR, un événement utilisateur est superflu si le code s'exécute dans le sandbox de l'application pour accéder au Presse-papiers.

Utilisation de la classe Capabilities

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Capabilities permet aux développeurs de déterminer l'environnement dans lequel s'exécute l'application. A l'aide des diverses propriétés de la classe Capabilities, vous pouvez déterminer la résolution et la langue du système de l'utilisateur, savoir si ce système prend en charge les logiciels d'accessibilité et identifier la version du moteur d'exécution Flash actuellement installée.

La vérification des propriétés de la classe Capabilities vous autorise à personnaliser votre application pour un fonctionnement optimal sur l'environnement de l'utilisateur. Par exemple, si vous vérifiez les propriétés `Capabilities.screenResolutionX` et `Capabilities.screenResolutionY`, vous pouvez déterminer la résolution d'affichage du système de l'utilisateur et décider de la taille de vidéo la plus appropriée. Vous pouvez aussi vérifier la propriété `Capabilities.hasMP3` pour voir si le système de l'utilisateur prend en charge la lecture du format mp3 avant d'essayer de charger un fichier mp3 externe.

Le code ci-après utilise une expression régulière pour analyser la version du moteur d'exécution Flash utilisée sur le client :

```
var versionString:String = Capabilities.version;
var pattern:RegExp = /^(\w*) (\d*),(\d*),(\d*),(\d*)$/;
var result:Object = pattern.exec(versionString);
if (result != null)
{
    trace("input: " + result.input);
    trace("platform: " + result[1]);
    trace("majorVersion: " + result[2]);
    trace("minorVersion: " + result[3]);
    trace("buildNumber: " + result[4]);
    trace("internalBuildNumber: " + result[5]);
}
else
{
    trace("Unable to match RegExp.");
}
```

Si vous souhaitez envoyer les capacités du système de l'utilisateur à un script côté serveur afin de stocker les informations dans une base de données, vous pouvez utiliser le code ActionScript suivant :

```
var url:String = "log_visitor.cfm";
var request:URLRequest = new URLRequest(url);
request.method = URLRequestMethod.POST;
request.data = new URLVariables(Capabilities.serverString);
var loader:URLLoader = new URLLoader(request);
```

Exemple d'utilisation de Capabilities : détection des capacités du système

Flash Player 9 et les versions ultérieures

L'exemple CapabilitiesExplorer vous montre comment utiliser la classe flash.system.Capabilities pour déterminer les fonctions prises en charge par la version du moteur d'exécution Flash de l'utilisateur. Cet exemple étudie les techniques suivantes :

- Détection des fonctions prises en charge par la version du moteur d'exécution Flash de l'utilisateur à l'aide de la classe Capabilities
- Utilisation de la classe ExternalInterface pour détecter les paramètres de navigation pris en charge par le navigateur de l'utilisateur

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application CapabilitiesExplorer se trouvent dans le dossier Samples/CapabilitiesExplorer. Cette application se compose des fichiers suivants :

Fichier	Description
CapabilitiesExplorer.fla ou CapabilitiesExplorer.mxml	Fichier d'application principal dans Flash (FLA) ou Flex (MXML).
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	Classe fournissant la principale fonctionnalité de l'application, notamment l'ajout des capacités du système dans un tableau, le tri des éléments et l'extraction des capacités du navigateur à l'aide de la classe ExternalInterface.
capabilities.html	Conteneur HTML comprenant le code JavaScript nécessaire à la communication avec l'API externe.

Présentation de CapabilitiesExplorer

Flash Player 9 et les versions ultérieures

Le fichier CapabilitiesExplorer.mxml se charge de définir l'interface utilisateur de l'application CapabilitiesExplorer. Les capacités de la version du moteur d'exécution Flash de l'utilisateur sont affichées dans une occurrence du composant DataGrid sur la scène. Les capacités du navigateur sont également affichées si l'application est exécutée à partir d'un conteneur HTML et si l'API externe est disponible.

Une fois que l'événement `creationComplete` du fichier de l'application principale est distribué, la méthode `initApp()` est appelée. La méthode `initApp()` appelle la méthode `getCapabilities()` à partir de la classe com.example.programmingas3.capabilities.CapabilitiesGrabber. Le code de la méthode `initApp()` se présente comme suit :

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

La méthode `CapabilitiesGrabber.getCapabilities()` renvoie un tableau trié contenant les capacités du moteur d'exécution Flash et du navigateur, qui est alors affecté à la propriété `dataProvider` de l'occurrence du composant DataGrid `capabilitiesGrid` sur la scène.

Présentation de la classe CapabilitiesGrabber

Flash Player 9 et les versions ultérieures

La méthode statique `getCapabilities()` de la classe CapabilitiesGrabber ajoute chaque propriété de la classe flash.system.Capabilities dans un tableau (`capDP`). Elle appelle ensuite la méthode statique `getBrowserObjects()` de la classe CapabilitiesGrabber. La méthode `getBrowserObjects()` utilise l'API externe pour passer en boucle sur l'objet navigator du navigateur, qui contient les capacités du navigateur. La méthode `getCapabilities()` se présente comme suit :

```
public static function getCapabilities():Array
{
    var capDP:Array = new Array();
    capDP.push({name:"Capabilities.avHardwareDisable", value:Capabilities.avHardwareDisable});
    capDP.push({name:"Capabilities.hasAccessibility", value:Capabilities.hasAccessibility});
    capDP.push({name:"Capabilities.hasAudio", value:Capabilities.hasAudio});
    ...
    capDP.push({name:"Capabilities.version", value:Capabilities.version});
    var navArr:Array = CapabilitiesGrabber.getBrowserObjects();
    if (navArr.length > 0)
    {
        capDP = capDP.concat(navArr);
    }
    capDP.sortOn("name", Array.CASEINSENSITIVE);
    return capDP;
}
```

La méthode `getBrowserObjects()` renvoie un tableau de toutes les propriétés de l'objet navigator du navigateur. Si ce tableau contient un élément ou plus, le tableau des capacités du navigateur (`navArr`) s'ajoute au tableau des capacités de Flash Player (`capDP`) et le tableau résultant est trié par ordre alphabétique. Enfin, le tableau trié est renvoyé au fichier de l'application principale, qui ensuite remplit la grille de données. Le code de la méthode `getBrowserObjects()` se présente comme suit :

```
private static function getBrowserObjects():Array
{
    var itemArr:Array = new Array();
    var itemVars:URLVariables;
    if (ExternalInterface.available)
    {
        try
        {
            var tempStr:String = ExternalInterface.call("JS_getBrowserObjects");
            itemVars = new URLVariables(tempStr);
            for (var i:String in itemVars)
            {
                itemArr.push({name:i, value:itemVars[i]});
            }
        }
        catch (error:SecurityError)
        {
            // ignore
        }
    }
    return itemArr;
}
```

Si l'API externe est disponible dans l'environnement de l'utilisateur, le moteur d'exécution Flash appelle la méthode JavaScript `JS_getBrowserObjects()`, qui passe en boucle sur l'objet navigator du navigateur et renvoie une chaîne de valeurs au format URL à ActionScript. Cette chaîne est alors convertie en un objet URLVariables (`itemVars`) et ajoutée au tableau `itemArr`, qui est renvoyé au script appelant.

Communication avec JavaScript

Flash Player 9 et les versions ultérieures

La dernière étape dans la construction de l'application CapabilitiesExplorer consiste à écrire le code JavaScript nécessaire au passage en boucle de chaque élément de l'objet navigator du navigateur et à l'ajout d'une paire nom-valeur dans un tableau temporaire. Le code de la méthode JavaScript `JS_getBrowserObjects()` dans le fichier container.html est le suivant :

```
<script language="JavaScript">
    function JS_getBrowserObjects()
    {
        // Create an array to hold each of the browser's items.
        var tempArr = new Array();

        // Loop over each item in the browser's navigator object.
        for (var name in navigator)
        {
            var value = navigator[name];

            // If the current value is a string or Boolean object, add it to the
            // array, otherwise ignore the item.
            switch (typeof(value))
            {
                case "string":
                case "boolean":

                    // Create a temporary string which will be added to the array.
                    // Make sure that we URL-encode the values using JavaScript's
                    // escape() function.
                    var tempStr = "navigator." + name + "=" + escape(value);
                    // Push the URL-encoded name/value pair onto the array.
                    tempArr.push(tempStr);
                    break;
            }
        }
        // Loop over each item in the browser's screen object.
        for (var name in screen)
        {
            var value = screen[name];

            // If the current value is a number, add it to the array, otherwise
            // ignore the item.
            switch (typeof(value))
            {
                case "number":
                    var tempStr = "screen." + name + "=" + escape(value);
                    tempArr.push(tempStr);
                    break;
            }
        }
        // Return the array as a URL-encoded string of name-value pairs.
        return tempArr.join("&");
    }
</script>
```

Le code commence par créer un tableau temporaire qui contiendra toutes les paires nom-valeur de l'objet navigator. Une boucle `for..in` est ensuite appliquée à l'objet navigator, et le type de données de la valeur actuelle est évaluée pour éliminer les valeurs indésirables. Dans cette application, seules les valeurs String ou Boolean nous intéressent. Les autres types de données (par exemple les fonctions ou les tableaux) sont ignorés. Chaque valeur String ou Boolean de l'objet navigator est ajoutée au tableau `tempArr`. Une boucle `for..in` est ensuite appliquée à l'objet screen du navigateur, et chaque valeur numérique est ajoutée au tableau `tempArr`. Enfin, le tableau temporaire est converti en chaîne à l'aide de la méthode `Array.join()`. Ce tableau utilise l'esperluette (&) comme délimiteur, ce qui permet à ActionScript d'analyser facilement les données à l'aide de la classe URLVariables.

Chapitre 50 : Appel et fermeture d'une application AIR

Adobe AIR 1.0 et les versions ultérieures

Cette section est consacrée aux différentes techniques d'appel d'une application Adobe® AIR® installée, ainsi qu'aux options et considérations de fermeture d'une application en cours d'exécution.

Remarque : *les objets NativeApplication, InvokeEvent et BrowserInvokeEvent ne sont proposés qu'au contenu SWF qui s'exécute dans le sandbox d'une application AIR. Un contenu SWF qui s'exécute dans le moteur d'exécution Flash Player, au sein du navigateur ou du lecteur autonome (projection), ou dans une application AIR hors du sandbox d'application, ne peut pas accéder à ces classes.*

Pour obtenir une explication rapide de l'ouverture et de la fermeture d'une application AIR et des exemples de code correspondants, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Options de démarrage

Voir aussi

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

Appel d'une application

Adobe AIR 1.0 et les versions ultérieures

Une application AIR est appelée lorsque l'utilisateur (ou le système d'exploitation) exécute l'une des actions suivantes :

- Il lance l'application à partir du shell de poste de travail.
- Il utilise l'application comme une commande sur un shell de ligne de commande.
- Il ouvre un type de fichier pour lequel cette application correspond à l'application d'ouverture définie par défaut.
- (Mac OS X) Il clique sur l'icône de l'application sur la barre des tâches du Dock (que l'application soit en cours d'exécution ou non).
- Il choisit de lancer l'application à partir du programme d'installation (à la fin d'un nouveau processus d'installation ou après un double-clic sur le fichier AIR d'une application déjà installée).
- Il commence une mise à jour d'une application AIR alors que la version installée l'informe qu'elle est déjà en train de traiter des mises à jour (par l'inclusion de la déclaration `<customUpdateUI>true</customUpdateUI>` dans le fichier descripteur d'application).
- (iOS) Il reçoit une notification du service APN (Apple Push Notification).
- Il invoque l'application via une URL.

- Il consulte une page Web hébergeant une application ou un badge Flash appelant la méthode `com.adobe.air.AIR launchApplication()` qui spécifie les informations d'identification relatives à l'application AIR. (Le descripteur d'application doit également comprendre une déclaration `<allowBrowserInvocation>true</allowBrowserInvocation>` afin que les appels au navigateur aboutissent.)

Dès qu'une application AIR est appelée, AIR distribue un objet InvokeEvent de type `invoke` par le biais de l'objet NativeApplication Singleton. Afin de laisser à une application le temps de s'initialiser et d'enregistrer un écouteur d'événement, les événements `invoke` sont placés en file d'attente au lieu d'être ignorés. Dès qu'un écouteur est enregistré, tous les événements placés en file d'attente sont livrés.

Remarque : *lorsqu'une application est appelée au moyen de la fonction d'appel du navigateur, l'objet NativeApplication distribue un événement `invoke` uniquement si l'application n'est pas en cours d'exécution.*

Pour recevoir des événements `invoke`, appelez la méthode `addEventListener()` de l'objet NativeApplication (`NativeApplication.nativeApplication`). Lorsqu'un écouteur d'événement s'enregistre pour un événement `invoke`, il reçoit également tous les événements `invoke` survenus avant l'enregistrement. Les événements `invoke` placés en file d'attente sont distribués un par un dans un court laps de temps après le renvoi de l'appel à `addEventListener()`. Si un nouvel événement `invoke` se produit au cours de ce processus, il peut être distribué avant un ou plusieurs des événements placés en file d'attente. Ce placement des événements en file d'attente vous permet de gérer tous les événements `invoke` survenus avant l'exécution de votre code d'initialisation. Sachez que si, plus tard dans l'exécution (après l'initialisation de l'application), vous ajoutez un écouteur d'événement, celui-ci recevra toujours tous les événements `invoke` survenus depuis le lancement de l'application.

Une seule occurrence d'une application AIR est lancée. Lorsqu'une application en cours d'exécution est à nouveau appelée, AIR distribue un nouvel événement `invoke` vers elle. C'est l'application AIR qui est chargée de répondre à un événement `invoke` et de prendre les mesures appropriées (comme l'ouverture d'une nouvelle fenêtre de document).

Un objet `InvokeEvent` contient tous les arguments transmis à l'application, de même qu'un répertoire à partir duquel l'application a été appelée. Si l'application a été appelée via une association de type de fichier, le chemin d'accès complet au fichier est alors inclus dans les arguments de ligne de commande. De la même manière, si l'application a été appelée via une mise à jour, le chemin d'accès complet au fichier AIR de mise à jour est fourni.

Lorsque plusieurs fichiers sont ouverts simultanément, un seul objet InvokeEvent est distribué sous Mac OS X. Chaque fichier est inclus dans le tableau `arguments`. Sous Windows et Linux, un objet InvokeEvent distinct est distribué pour chaque fichier.

Votre application gère les événements `invoke` en enregistrant un écouteur avec son objet NativeApplication :

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

Et en définissant un écouteur d'événement :

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

Capture des arguments de ligne de commande

Adobe AIR 1.0 et les versions ultérieures

Les arguments de ligne de commande associés à l'appel d'une application AIR sont livrés dans l'objet InvokeEvent distribué par l'objet NativeApplication. La propriété `arguments` d'un objet InvokeEvent contient un tableau des arguments transmis par le système d'exploitation suite à l'appel d'une application AIR. Si les arguments contiennent des chemins de fichiers relatifs, il est généralement possible de résoudre les chemins à l'aide de la propriété `currentDirectory`.

Les arguments transmis à un programme AIR sont traités sous forme de chaînes délimitées par des espaces, à moins d'être placés entre guillemets doubles :

Arguments	Array
tick tock	{tick,tock}
tick "tick tock"	{tick,tick tock}
"tick" “tock”	{tick,tock}
\"tick\" \"tock\"	{"tick","tock"}

La propriété `currentDirectory` d'un objet InvokeEvent contient un objet File représentant le répertoire à partir duquel l'application a été lancée.

Lorsqu'une application est appelée suite à l'ouverture d'un fichier dont le type est enregistré par l'application, le chemin d'accès natif au fichier est compris dans les arguments de ligne de commande sous forme de chaîne. (Votre application est chargée d'ouvrir le fichier ou d'effectuer l'opération attendue.) De la même manière, lorsqu'une application est programmée pour se mettre à jour automatiquement (plutôt que de dépendre de l'interface utilisateur de mise à jour d'AIR standard), le chemin d'accès natif au fichier AIR est inclus si un utilisateur double-clique sur un fichier AIR contenant une application dotée de l'ID d'application correspondant.

Vous pouvez accéder au fichier à l'aide de la méthode `resolve()` de l'objet File `currentDirectory` :

```
if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

Il est également recommandé de vérifier qu'un argument correspond effectivement à un chemin d'accès à un fichier.

Exemple : Journal des événements d'appel

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant explique comment enregistrer des écouteurs pour l'événement `invoke` et comment gérer ce type d'événement. Dans cet exemple, tous les événements d'appel reçus sont consignés dans un journal et le répertoire actif ainsi que les arguments de ligne de commande sont affichés.

Exemple ActionScript

```
package
{
    import flash.display.Sprite;
    import flash.events.InvokeEvent;
    import flash.desktop.NativeApplication;
    import flash.text.TextField;

    public class InvokeEventLogExample extends Sprite
    {
        public var log:TextField;

        public function InvokeEventLogExample()
        {
            log = new TextField();
            log.x = 15;
            log.y = 15;
            log.width = 520;
            log.height = 370;
            log.background = true;

            addChild(log);

            NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvoke);
        }

        public function onInvoke(invokeEvent:InvokeEvent):void
        {
            var now:String = new Date().toTimeString();
            logEvent("Invoke event received: " + now);

            if (invokeEvent.currentDirectory != null)
            {
                logEvent("Current directory=" + invokeEvent.currentDirectory.nativePath);
            }
            else
            {
```

```
                logEvent("--no directory information available--");
            }

            if (invokeEvent.arguments.length > 0)
            {
                logEvent("Arguments: " + invokeEvent.arguments.toString());
            }
            else
            {
                logEvent("--no arguments--");
            }
        }

        public function logEvent(entry:String):void
        {
            log.appendText(entry + "\n");
            trace(entry);
        }
    }
}
```

Exemple Flex

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
    invoke="onInvoke(event)" title="Invocation Event Log">
    <mx:Script>
    <![CDATA[
    import flash.events.InvokeEvent;
    import flash.desktop.NativeApplication;

    public function onInvoke(invokeEvent:InvokeEvent):void {
        var now:String = new Date().toTimeString();
        logEvent("Invoke event received: " + now);

        if (invokeEvent.currentDirectory != null){
            logEvent("Current directory=" + invokeEvent.currentDirectory.nativePath);
        } else {
            logEvent("--no directory information available--");
        }

        if (invokeEvent.arguments.length > 0){
            logEvent("Arguments: " + invokeEvent.arguments.toString());
        } else {
            logEvent("--no arguments--");
        }
    }

    public function logEvent(entry:String):void {
        log.text += entry + "\n";
        trace(entry);
    }
    ]]>
    </mx:Script>
    <mx:TextArea id="log" width="100%" height="100%" editable="false"
        valueCommit="log.verticalScrollPosition=log.textHeight;"/>
</mx:WindowedApplication>
```

Appel d'une application AIR lors de la connexion d'un utilisateur

Adobe AIR 1.0 et les versions ultérieures

Il est possible de configurer le lancement automatique d'une application AIR au moment de la connexion de l'utilisateur actif en définissant la propriété NativeApplication `startAtLogin` sur `true`. Une fois ce paramètre défini, l'application est lancée automatiquement chaque fois que l'utilisateur se connecte. Elle continue à démarrer lors d'une connexion tant que le paramètre n'est pas défini sur `false`. Soit ce changement est effectué manuellement par l'utilisateur via le système d'exploitation, soit il survient suite à la désinstallation de l'application. Le lancement au moment de la connexion est un paramètre d'exécution. Le paramètre s'applique uniquement à l'utilisateur actif. L'application doit être installée pour que la propriété `startAtLogin` soit définie sur `true`. Une erreur est renvoyée si la propriété est définie alors que l'application n'est pas installée (suite à un lancement à l'aide d'ADL, par exemple).

Remarque : *l'application n'est pas lancée au démarrage du système informatique. Elle s'ouvre lorsque l'utilisateur se connecte.*

Pour déterminer si une application a démarré automatiquement ou si son lancement résulte d'une action utilisateur, vous pouvez examiner la propriété `reason` de l'objet InvokeEvent. Si la propriété est définie sur `InvokeEventReason.LOGIN`, l'application a démarré automatiquement. Pour d'autres chemins d'invocation, la propriété `reason` est définie comme suit :

- `InvokeEventReason.NOTIFICATION` (iOS uniquement) : l'application a été invoquée via le service APN. Pour plus d'informations sur le service APN, voir Utilisation de notifications Push.
- `InvokeEventReason.OPEN_URL` : l'application a été invoquée par une autre application ou par le système.
- `InvokeEventReason.Standard` : tous les autres cas.

Pour accéder à la propriété `reason`, votre application doit cibler AIR 1.5.1 ou une version ultérieure (pour ce faire, définissez la valeur d'espace de noms correcte dans le fichier descripteur de l'application).

L'application simplifiée suivante fait appel à la propriété reason de l'objet InvokeEvent pour déterminer son comportement lorsqu'il se produit un événement invoke. Si la propriété reason est définie sur « login », l'application demeure en arrière-plan. Si tel n'est pas le cas, l'application principale devient visible. Une application qui fait appel à cette technique démarre généralement lorsque l'utilisateur se connecte pour pouvoir exécuter un traitement en arrière-plan ou une surveillance d'événement et ouvre une fenêtre en réponse à un événement déclenché par l'utilisateur.

```
package {
    import flash.desktop.InvokeEventReason;
    import flash.desktop.NativeApplication;
    import flash.display.Sprite;
    import flash.events.InvokeEvent;

    public class StartAtLogin extends Sprite
    {
        public function StartAtLogin()
        {
            try
            {
                NativeApplication.nativeApplication.startAtLogin = true;
            }
            catch ( e:Error )
            {
                trace( "Cannot set startAtLogin:" + e.message );
            }

            NativeApplication.nativeApplication.addEventListener( InvokeEvent.INVOKE, onInvoke );
        }

        private function onInvoke( event:InvokeEvent ):void
        {
            if( event.reason == InvokeEventReason.LOGIN )
            {
                //do background processing...
                trace( "Running in background..." );
            }
            else
            {
                this.stage.nativeWindow.activate();
            }
        }
    }
}
```

Remarque : *pour évaluer la différence de comportement, mettez en package l'application et installez-la. La propriété* `startAtLogin` *est réservée aux applications installées.*

Appel d'une application AIR à partir du navigateur

Adobe AIR 1.0 et les versions ultérieures

La fonction d'appel du navigateur permet à un site Web de lancer une application AIR installée à partir du navigateur. L'appel du navigateur est uniquement autorisé si le fichier descripteur d'application définit `allowBrowserInvocation` sur `true` :

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Lorsque l'application est appelée par le biais du navigateur, l'objet NativeApplication de l'application distribue un objet BrowserInvokeEvent.

Pour recevoir des événements BrowserInvokeEvent, appelez la méthode `addEventListener()` de l'objet NativeApplication (`NativeApplication.nativeApplication`) dans l'application AIR. Lorsqu'un écouteur d'événement s'enregistre pour un événement BrowserInvokeEvent, il reçoit également tous les événements BrowserInvokeEvent survenus avant l'enregistrement. Ces événements sont distribués après le renvoi de l'appel à `addEventListener()`, mais pas nécessairement avant d'autres événements BrowserInvokeEvent susceptibles d'être reçus après l'enregistrement. Cela vous permet de gérer les événements BrowserInvokeEvent survenus avant l'exécution de votre code d'initialisation (comme, par exemple, lorsque l'application a été appelée au départ depuis le navigateur). Sachez que si, plus tard dans l'exécution (après l'initialisation de l'application), vous ajoutez un écouteur d'événement, celui-ci recevra toujours tous les événements BrowserInvokeEvent survenus depuis le lancement de l'application.

L'objet BrowserInvokeEvent comprend les propriétés suivantes :

Propriété	**Description**
arguments	Tableau d'arguments (de chaînes) à transmettre à l'application.
isHTTPS	Indique si le contenu du navigateur utilise le modèle d'URL https (`true`) ou pas (`false`).
isUserEvent	Indique si l'appel du navigateur a entraîné un événement utilisateur (tel qu'un clic de souris). Dans AIR 1.0, cette propriété est toujours définie sur `true` ; AIR requiert un événement utilisateur pour la fonction d'appel du navigateur.
sandboxType	Type de sandbox relatif au contenu du navigateur. Les valeurs valides sont identiques à celles qui sont admises pour la propriété `Security.sandboxType`. Il peut s'agir de l'une des valeurs suivantes : • `Security.APPLICATION` : le contenu se trouve dans le sandbox de sécurité de l'application. • `Security.LOCAL_TRUSTED` : le contenu se trouve dans le sandbox de sécurité local avec système de fichiers. • `Security.LOCAL_WITH_FILE` : le contenu se trouve dans le sandbox de sécurité local avec système de fichiers. • `Security.LOCAL_WITH_NETWORK` : le contenu se trouve dans le sandbox de sécurité local avec accès au réseau. • `Security.REMOTE` : le contenu se trouve dans un domaine (réseau) distant.
securityDomain	Correspond au domaine de sécurité du contenu du navigateur, tel `www.adobe.com` ou `www.example.org`. Cette propriété est uniquement définie pour le contenu du sandbox de sécurité distant (pour le contenu d'un domaine réseau). Elle n'est pas définie pour un contenu figurant dans un sandbox de sécurité d'application ou local.

Si vous utilisez la fonction d'appel du navigateur, tenez compte des implications au niveau de la sécurité. Lorsqu'un site Web lance une application AIR, il peut envoyer les données par le biais de la propriété `arguments` de l'objet BrowserInvokeEvent. Utilisez ces données avec précaution dans toute opération délicate, telle que des API de chargement de code ou de fichier. Le niveau de risque varie en fonction de l'usage que l'application réserve aux données. Si seul un site Web spécifique doit appeler l'application, celle-ci doit vérifier la propriété `securityDomain` de l'objet BrowserInvokeEvent. Vous pouvez également exiger de la part du site Web appelant l'application qu'il utilise le protocole HTTPS, ce que vous pouvez contrôler en vérifiant la propriété `isHTTPS` de l'objet BrowserInvokeEvent.

L'application devrait valider les données transmises. Si, par exemple, une application s'attend à recevoir des URL pointant vers un domaine spécifique, elle devrait vérifier que les URL pointent réellement vers le domaine attendu. Cette procédure peut empêcher un pirate de tromper l'application en lui demandant de lui envoyer des données sensibles.

Aucune application ne devrait utiliser d'arguments BrowserInvokeEvent susceptibles de pointer vers des ressources locales. Ainsi, il est vivement déconseillé de concevoir une application qui crée des objets File à partir d'un chemin transmis depuis le navigateur. Si le navigateur peut transmettre des chemins distants, l'application devrait vérifier que ces derniers n'utilisent pas le protocole `file://` à la place d'un protocole distant.

Fermeture d'une application

Adobe AIR 1.0 et les versions ultérieures

Le moyen le plus rapide pour fermer une application consiste à appeler la méthode NativeApplication exit(). Cette approche fonctionne correctement si votre application ne doit pas enregistrer de données ou nettoyer une ressource externe. L'appel de la méthode `exit()` entraîne la fermeture de toutes les fenêtres, puis celle de l'application. Toutefois, pour permettre aux fenêtres ou à d'autres composants de l'application d'interrompre le processus de fermeture, afin d'enregistrer des données cruciales par exemple, distribuez les événements d'avertissement pertinents avant d'appeler `exit()`.

Un autre point à prendre en compte dans le cadre de l'arrêt progressif d'une application est l'utilisation d'un seul chemin d'exécution, quelle que soit la manière dont le processus d'arrêt commence. L'utilisateur (ou le système d'exploitation) peut déclencher la fermeture de l'application de l'une des manières suivantes :

- En fermant la dernière fenêtre de l'application lorsque la méthode `NativeApplication.nativeApplication.autoExit` est définie sur `true`.
- En sélectionnant la commande de fermeture d'application à partir du système d'exploitation comme, par exemple, lorsque l'utilisateur choisit la commande Quitter de l'application dans le menu par défaut. (Cela se produit uniquement sous Mac OS, car Windows et Linux ne fournissent pas de commande de fermeture d'application via le chrome système.)
- En arrêtant l'ordinateur.

Lorsqu'une commande de fermeture est traitée par le biais du système d'exploitation selon l'une de ces méthodes, NativeApplication distribue un événement `exiting`. Si aucun écouteur n'annule l'événement `exiting`, toutes les fenêtres qui étaient ouvertes se ferment. Chaque fenêtre distribue un événement `closing` suivi d'un événement `close`. Si l'une des fenêtres annule l'événement `closing`, le processus d'arrêt est interrompu.

Si l'ordre de fermeture des fenêtres présente un problème pour votre application, écoutez l'événement `exiting` de NativeApplication et fermez manuellement les fenêtres dans l'ordre approprié. Tel est par exemple le cas lorsqu'une fenêtre de document contient des palettes d'outils. Il peut s'avérer peu pratique (voire pire) que le système ferme les palettes alors que l'utilisateur a choisi d'annuler la commande de fermeture afin d'enregistrer des données. Sous Windows, vous obtiendrez uniquement l'événement `exiting` après la fermeture de la dernière fenêtre (lorsque la propriété `autoExit` de l'objet NativeApplication est définie sur `true`).

Pour adopter un comportement homogène sur toutes les plates-formes, que la séquence de fermeture soit lancée via le chrome du système d'exploitation, les commandes de menu ou la logique de l'application, suivez les recommandations ci-après pour fermer l'application :

1 Distribuez toujours un événement `exiting` par le biais de l'objet NativeApplication avant d'appeler `exit()` dans le code de l'application et vérifiez qu'aucun autre composant de l'application n'annule l'événement.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2 Ecoutez l'événement `exiting` de l'application à partir de l'objet `NativeApplication.nativeApplication` et, dans le gestionnaire, fermez toutes les fenêtres ouvertes (en distribuant d'abord un événement `closing`). Effectuez les éventuelles tâches de nettoyage nécessaires (enregistrement des données de l'application, suppression des fichiers temporaires, etc.) une fois toutes les fenêtres fermées. Faites exclusivement appel à des méthodes synchrones lors du nettoyage afin de vous assurer qu'elles sont bien terminées avant la fermeture de l'application.

Si l'ordre de fermeture des fenêtres est sans importance, vous pouvez analyser en boucle le tableau `NativeApplication.nativeApplication.openedWindows` et fermer chaque fenêtre une après l'autre. Si l'ordre de fermeture *compte*, élaborez un moyen de l'appliquer aux fenêtres.

```
private function onExiting(exitingEvent:Event):void {
    var winClosingEvent:Event;
    for each (var win:NativeWindow in NativeApplication.nativeApplication.openedWindows) {
        winClosingEvent = new Event(Event.CLOSING,false,true);
        win.dispatchEvent(winClosingEvent);
        if (!winClosingEvent.isDefaultPrevented()) {
            win.close();
        } else {
            exitingEvent.preventDefault();
        }
    }

    if (!exitingEvent.isDefaultPrevented()) {
        //perform cleanup
    }
}
```

3 Les fenêtres devraient toujours gérer leur propre nettoyage en écoutant les événements `closing` qui les concernent.

4 Utilisez un seul écouteur `exiting` dans l'application, car les gestionnaires appelés auparavant ignorent si des gestionnaires ultérieurs annuleront l'événement `exiting` (sans compter qu'il serait déraisonnable de se fier à l'ordre d'exécution).

Chapitre 51 : Utilisation des informations sur le moteur d'exécution d'AIR et les systèmes d'exploitation

Adobe AIR 1.0 et les versions ultérieures

La présente section décrit comment une application AIR peut gérer des associations de fichiers de systèmes d'exploitation, constater les activités des utilisateurs et lire des informations sur le moteur d'exploitation Adobe® AIR®.

Voir aussi

flash.desktop.NativeApplication

Gestion des associations de fichiers

Adobe AIR 1.0 et les versions ultérieures

Les associations entre votre application et un type de fichier doivent être déclarées dans le descripteur d'application. Au cours du processus d'installation, le programme d'installation de l'application AIR spécifie celle-ci comme application de démarrage par défaut pour chacun des types de fichiers déclarés, à moins qu'une autre application ne le soit déjà par défaut. Le processus d'installation de l'application AIR n'écrase pas une association de types de fichiers existants. Pour remplacer l'association par une autre application, appelez la méthode `NativeApplication.setAsDefaultApplication()` lors de l'exécution.

Il est recommandé de s'assurer que les associations de fichiers prévues sont en place lorsque votre application démarre. Ceci, parce que le programme d'installation de l'application AIR n'annule pas les associations de fichiers existantes et que ces associations sur un système d'utilisateur peuvent changer à tout moment. Lorsqu'une autre application dispose de l'association de fichiers actuelle, il est recommandé par courtoisie de demander une autorisation à l'utilisateur avant de prendre le contrôle de l'association en cours.

Les méthodes suivantes de la classe NativeApplication permettent à une application de gérer des associations de fichiers. Chacune des méthodes prend l'extension du type de fichier comme paramètre.

Méthode	Description
isSetAsDefaultApplication()	Renvoie true si l'application AIR est actuellement associée au type de fichier spécifié.
setAsDefaultApplication()	Crée l'association entre l'application AIR et l'opération d'ouverture du type de fichier.
removeAsDefaultApplication()	Annule l'association entre l'application AIR et le type de fichier.
getDefaultApplication()	Signale le chemin de l'application qui est actuellement associée au type de fichier.

AIR ne peut gérer des associations que pour des types de fichiers déclarés à l'origine dans le descripteur d'application. Il n'est pas possible de lire des informations sur les associations d'un type de fichier non déclaré, même si un utilisateur a créé manuellement l'association entre ce type de fichier et votre application. L'appel de toute méthode de gestion d'association de fichiers avec l'extension, pour un type de fichier non déclaré dans le descripteur d'application, provoque le renvoi d'une exception d'exécution par l'application

Lecture de la version du moteur d'exécution et du correctif

Adobe AIR 1.0 et les versions ultérieures

L' objet NativeApplication possède une propriété `runtimeVersion`, qui correspond à la version du moteur d'exécution dans lequel l'application est exécutée (une chaîne telle que `"1.0.5"`). L'objet NativeApplication possède également une propriété `runtimePatchLevel` qui est un niveau du correctif pour le moteur d'exécution, un nombre tel que 2960. Le code ci-dessous utilise ces propriétés :

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

Détection des capacités d'AIR

Adobe AIR 1.0 et les versions ultérieures

Pour un fichier regroupé avec l'application Adobe AIR, la propriété `Security.sandboxType` est définie sur la valeur spécifiée par la constante `Security.APPLICATION`. Vous pouvez charger le contenu, qui peut contenir ou non des interfaces de programmation spécifiques à AIR, selon qu'un fichier se trouve ou non dans le sandbox de sécurité d'Adobe Air, comme le montre le code ci-dessous :

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

Toutes les ressources qui ne sont pas installées avec l'application AIR sont affectées aux mêmes sandbox de sécurité que ceux qu'Adobe® Flash® Player auraient affecté dans un navigateur Web. Les ressources distantes sont placées dans des sandbox selon leurs domaines d'origine tandis que les ressources locales le sont dans un sandbox local avec accès au réseau, local avec système de fichiers ou approuvé localement.

Vous pouvez contrôler si la propriété statique `Capabilities.playerType` est définie sur `"Desktop"` pour voir si le contenu est exécuté dans le moteur d'exécution (et non pas dans Flash Player qui, lui, est exécuté dans un navigateur).

Pour plus d'informations, voir « Sécurité AIR » à la page 1122.

Suivi de la présence des utilisateurs

Adobe AIR 1.0 et les versions ultérieures

L'objet NativeApplication distribue deux événements qui vous aident à détecter à quel moment un utilisateur est actif sur son ordinateur. Si aucune activité de souris ou de clavier n'est détectée dans l'intervalle fixé par la propriété `NativeApplication.idleThreshold`, la NativeApplication distribue un événement `userIdle`. Lorsque survient l'entrée suivante par le clavier ou par la souris, l'objet NativeApplication distribue un événement `userPresent`. L'intervalle `idleThreshold` est mesuré en secondes et il a une valeur par défaut de 300, soit cinq minutes. Vous pouvez également lire le nombre de secondes depuis la dernière saisie de l'utilisateur grâce à la propriété `NativeApplication.nativeApplication.lastUserInput`.

Les lignes de code ci-dessous définissent le délai d'inactivité sur deux minutes et elles sont à l'écoute des deux événements `userIdle` et `userPresent` :

```
NativeApplication.nativeApplication.idleThreshold = 120;
NativeApplication.nativeApplication.addEventListener(Event.USER_IDLE, function(event:Event) {
    trace("Idle");
});
NativeApplication.nativeApplication.addEventListener(Event.USER_PRESENT,
function(event:Event) {
    trace("Present");
});
```

Remarque : *il n'y a qu'un seul événement* `userIdle` *distribué entre deux événements* `userPresent` *quels qu'ils soient.*

Chapitre 52 : Utilisation des fenêtres natives AIR

Adobe AIR 1.0 et les versions ultérieures

Utilisez les classes fournies par l'API de la fenêtre native d'Adobe® AIR® pour créer et gérer les fenêtres du poste de travail.

Principes de base des fenêtres natives dans AIR

Adobe AIR 1.0 et les versions ultérieures

Pour obtenir une explication rapide de l'utilisation des fenêtres natives dans AIR et des exemples de code correspondants, voir les articles de démarrage rapide suivants dans Adobe Developer Connection :

- Création d'une application munie de fenêtres transparentes (Flex)
- Interaction avec une fenêtre (Flex)
- Personnalisation de l'aspect d'une fenêtre native (Flex)
- Démarrage de fenêtres (Flex)
- Création de fenêtres affichées l'une derrière l'autre (Flex)
- Contrôle de l'ordre d'affichage des fenêtres (Flex)
- Création de fenêtres redimensionnables et non rectangulaires (Flex)
- Interaction avec une fenêtre (Flash)
- Personnalisation de l'aspect d'une fenêtre native (Flash)
- Création de fenêtres affichées l'une derrière l'autre (Flash)
- Contrôle de l'ordre d'affichage des fenêtres (Flash)
- Création de fenêtres redimensionnables et non rectangulaires (Flash)

AIR offre une API multiplateformes facile à utiliser vous permettant de créer des fenêtres natives dans un système d'exploitation à l'aide de Flash®, Flex™ et des techniques de programmation HTML.

Grâce à AIR, vous pouvez personnaliser l'aspect de votre application en toute liberté. Les fenêtres que vous créez peuvent ressembler à une application de bureau standard, en reproduisant le style Apple lors d'une exécution sous Mac, en respectant les conventions Microsoft lors d'une exécution sous Windows, et en s'harmonisant au gestionnaire de fenêtres sous Linux, sans inclure une seule ligne de code spécifique à une plate-forme. Vous pouvez par ailleurs utiliser le chrome extensible, auquel il est possible d'appliquer une enveloppe, fourni par la structure d'application Flex afin de créer votre propre style, quelle que soit la plate-forme d'exécution de votre application. Vous pouvez de surcroît concevoir votre propre chrome de fenêtre avec des vecteurs et des images bitmap, et utiliser les effets de transparence, ainsi que le fondu alpha avec le poste de travail. Vous en avez assez des fenêtres rectangulaires ? Créez des fenêtres arrondies !

Fenêtres dans AIR

Adobe AIR 1.0 et les versions ultérieures

AIR prend en charge trois API distinctes pour l'utilisation des fenêtres :

- La classe NativeWindow, orientée ActionScript, fournit l'API de la fenêtre du niveau le plus bas. Utilisez la classe NativeWindow dans les applications programmées dans ActionScript et Flash Professional. Pensez à étendre la classe NativeWindow pour spécialiser les fenêtres utilisées dans votre application.
- Dans l'environnement HTML, vous pouvez utiliser la classe Window de JavaScript, comme dans toute application Web à base de navigateur. Les appels aux méthodes Window de JavaScript sont transmis à l'objet window natif sous-jacent.
- Les classes Flex framework mx:WindowedApplication et mx:Window fournissent une « enveloppe » Flex à la classe NativeWindow. Le composant WindowedApplication remplace le composant Application lorsque vous créez une application AIR avec Flex et doit toujours être utilisé comme fenêtre initiale dans votre application Flex.

Fenêtres ActionScript

Lorsque vous créez des fenêtres avec la classe NativeWindow, utilisez directement la scène et la liste d'affichage de Flash Player. Pour ajouter un objet visuel à une classe NativeWindow, ajoutez l'objet à la liste d'affichage de la scène de la fenêtre ou à un autre conteneur d'objet d'affichage sur la scène.

Fenêtres HTML

Lorsque vous créez des fenêtres HTML, utilisez HTML, CSS et JavaScript pour afficher du contenu. Pour ajouter un objet visuel à une fenêtre HTML, ajoutez ce contenu au DOM HTML. Les fenêtres HTML sont une catégorie spéciale de l'élément NativeWindow. L'hôte AIR définit une propriété `nativeWindow` dans les fenêtres HTML, qui fournit l'accès à l'occurrence de NativeWindow sous-jacente. Utilisez cette propriété pour accéder aux propriétés, méthodes et événements de l'élément NativeWindow décrits ici.

Remarque : *l'objet Window de JavaScript dispose également de méthodes pour la rédaction de scripts dans la fenêtre conteneur, telles que* `moveTo()` *et* `close()`*. Lorsque plusieurs méthodes sont disponibles, utilisez celle qui vous convient le mieux.*

Fenêtres de la structure d'application Flex

Lorsque vous créez des fenêtres avec la structure d'application Flex, vous utilisez normalement des composants MXML pour remplir la fenêtre. Pour ajouter un composant Flex à une fenêtre, ajoutez l'élément de composant à la définition MXML de la fenêtre. Vous pouvez également utiliser ActionScript pour ajouter du contenu de façon dynamique. Les composants mx:WindowedApplication et mx:Window sont conçus comme conteneurs Flex et peuvent par conséquent accepter directement les composants Flex, ce que les objets NativeWindow ne permettent pas. Le cas échéant, il est possible d'accéder aux propriétés et méthodes NativeWindow par le biais des objets WindowedApplication et Window à l'aide de la propriété `nativeWindow`.

La fenêtre initiale de l'application

AIR crée pour vous la première fenêtre de l'application. AIR définit les propriétés et le contenu de la fenêtre à l'aide des paramètres spécifiés dans l'élément `initialWindow` du fichier descripteur d'application.

Si le contenu racine est un fichier SWF, AIR crée une occurrence de NativeWindow, charge le fichier SWF et l'ajoute à la scène de la fenêtre. Si le contenu racine est un fichier HTML, AIR crée une fenêtre HTML et charge le fichier HTML.

Classes d'une fenêtre native

Adobe AIR 1.0 et les versions ultérieures

L'API de la fenêtre native contient les classes suivantes :

Package	Classes
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

Flux d'événements des fenêtres natives

Adobe AIR 1.0 et les versions ultérieures

Les fenêtres natives distribuent des événements pour notifier les composants intéressés qu'un changement important est sur le point de se produire ou s'est déjà produit. La plupart des événements basés sur les fenêtres sont distribués par paires. Le premier événement informe qu'un changement est sur le point de se produire. Le deuxième événement annonce qu'un changement a été effectué. Vous pouvez annuler un événement d'avertissement, mais pas un événement de notification. La séquence suivante illustre le flux d'événements qui se produit lorsqu'un utilisateur clique sur le bouton d'agrandissement d'une fenêtre :

1 L'objet NativeWindow distribue un événement `displayStateChanging`.

2 Si aucun écouteur enregistré n'annule l'événement, la fenêtre est agrandie.

3 L'objet NativeWindow distribue un événement `displayStateChange`.

L'objet NativeWindow distribue en outre des événements lors de modifications associées ayant trait à la taille et à la position de la fenêtre. La fenêtre ne distribue aucun événement d'avertissement pour ces modifications associées. Les événements associés sont les suivants :

a Un événement `move` est distribué si le coin supérieur gauche de la fenêtre s'est déplacé suite à un agrandissement.

b Un événement `resize` est distribué si la taille de la fenêtre a changé suite à un agrandissement.

Un objet NativeWindow distribue une séquence d'événements similaire lors de l'agrandissement, de la restauration, de la fermeture, du déplacement et du redimensionnement d'une fenêtre.

Les événements d'avertissement ne sont distribués que lorsqu'une modification est initiée via le chrome de la fenêtre ou un autre mécanisme contrôlé par le système d'exploitation. Lorsque vous appelez une méthode window pour modifier la taille, la position ou l'état d'affichage de la fenêtre, la fenêtre ne distribue qu'un seul événement pour annoncer la modification. Vous pouvez distribuer un événement d'avertissement, si vous le souhaitez, à l'aide de la méthode `dispatchEvent()` de la fenêtre, puis vérifier si votre événement d'avertissement a été annulé avant d'effectuer la modification.

Pour plus d'informations sur les classes, les méthodes, les propriétés et les événements de l'API de fenêtre, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Propriétés de contrôle du style et du comportement d'une fenêtre native

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les propriétés suivantes contrôlent l'aspect et le comportement de base d'une fenêtre :

- `type`
- `systemChrome`
- `transparent`
- `owner`

Lorsque vous créez une fenêtre, vous définissez ces propriétés sur l'objet NativeWindowInitOptions transmis au constructeur de fenêtre. AIR lit les propriétés de la fenêtre d'application initiale à partir du fichier descripteur d'application. (A l'exception de la propriété `type` qui ne peut pas être définie dans le fichier descripteur d'application et dont la valeur est toujours `normal`.) Il est impossible de modifier les propriétés une fois la fenêtre créée.

Certains paramètres de ces propriétés sont mutuellement incompatibles : `systemChrome` ne peut pas être défini sur `standard` lorsque `transparent` est défini sur `true` ou `type` est défini sur `lightweight`.

Types de fenêtre

Adobe AIR 1.0 et les versions ultérieures

Les types de fenêtres d'AIR combinent les attributs de chrome et de visibilité du système d'exploitation natif pour créer trois types de fenêtre fonctionnels. Utilisez les constantes définies dans la classe NativeWindowType pour référencer les noms des types de fenêtre dans le code. AIR fournit les types de fenêtre suivants :

Type	Description
Normale	Une fenêtre classique. Les fenêtres normales utilisent le chrome plein écran et apparaissent dans la barre des tâches Windows et le menu de la fenêtre Mac OS X.
Utilitaire	Une palette d'outils. Les fenêtres d'utilitaire utilisent une version plus fine du chrome système et n'apparaissent ni dans la barre des tâches Windows ni dans le menu de la fenêtre Mac OS X.
Légère	Les fenêtres légères n'utilisent pas de chrome et n'apparaissent ni dans la barre des tâches Windows ni dans le menu de la fenêtre Mas OS X. De plus, les fenêtres légères ne disposent pas du menu Système (Alt-Espace) sous Windows. Les fenêtres légères sont adaptées aux commandes et aux bulles de notification (listes déroulantes qui ouvrent une zone d'affichage de courte durée, par exemple). Lorsque le `type` légère est utilisé, la propriété `systemChrome` doit être définie sur `none`.

Chrome de la fenêtre

Adobe AIR 1.0 et les versions ultérieures

Le chrome d'une fenêtre est un ensemble de commandes qui permet aux utilisateurs de manipuler une fenêtre dans l'environnement de bureau. Le chrome comprend les éléments suivants : barre de titre, boutons de la barre de titre, encadrement et poignées de redimensionnement.

Chrome système

Vous pouvez définir la propriété `systemChrome` sur `standard` ou sur `none`. Choisissez le chrome système `standard` pour attribuer à votre fenêtre le jeu de commandes standard créé par le système d'exploitation de l'utilisateur. Choisissez `none` pour créer votre propre chrome de fenêtre. Utilisez les constantes définies dans la classe NativeWindowSystemChrome pour référencer les paramètres du chrome système dans le code.

Le chrome système est géré par le système. Votre application ne bénéficie pas d'un accès direct aux commandes, mais peut réagir aux événements distribués lorsque les commandes sont utilisées. Lorsque vous utilisez le chrome standard pour une fenêtre, la propriété `transparent` doit être définie sur `false` et la propriété `type` doit être définie sur `normal` ou sur `utility`.

Chrome Flex

Lorsque vous utilisez les composants WindowedApplication ou Window de Flex, la fenêtre peut faire appel soit au chrome système, soit au chrome fourni par la structure d'application Flex. Pour utiliser le chrome Flex, définissez la propriété `systemChrome` requise pour créer la fenêtre sur `none`. Si vous utilisez les composants spark de Flex 4 au lieu des composants mx, vous devez spécifier la classe d'enveloppe pour faire appel au chrome de Flex. Libre à vous de faire appel aux enveloppes intégrées ou à vos propres enveloppes. L'exemple suivant illustre l'utilisation de la classe d'enveloppe WindowedApplication des composants spark intégrée pour proposer le chrome de fenêtre :

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/mx">
<fx:Style>
@namespace "library://ns.adobe.com/flex/spark";
WindowedApplication
{
    skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
}
</fx:Style>
</s:WindowedApplication>
```

Pour plus d'informations, voir Using Flex 4: About the AIR window containers: Controlling window chrome (disponible en anglais uniquement)

Chrome personnalisé

Lorsque vous créez une fenêtre sans chrome système, vous devez ajouter vos propres commandes de chrome afin de gérer les interactions entre un utilisateur et la fenêtre. Vous êtes également libre de créer des fenêtres transparentes et non rectangulaires.

Pour utiliser un chrome personnalisé avec le composant mx:WindowedApplication ou mx:Window, vous devez définir le style `showFlexChrome` sur `false`. Si tel n'est pas le cas, Flex intègre son propre chrome à vos fenêtres.

Transparence de la fenêtre

Adobe AIR 1.0 et les versions ultérieures

Pour permettre le fondu alpha d'une fenêtre avec le bureau ou d'autres fenêtres, définissez la propriété `transparent` de la fenêtre sur `true`. La propriété `transparent`, qui doit être définie avant de créer la fenêtre, ne peut pas être modifiée.

Une fenêtre transparente ne dispose d'aucun arrière-plan par défaut. Les zones d'une fenêtre qui ne contiennent aucun objet créé par l'application sont invisibles. Si le paramètre alpha d'un objet affiché est inférieur à un, tout élément affiché sous l'objet apparaît, notamment les autres objets d'affichage de la fenêtre, les autres fenêtres et le bureau.

Les fenêtres transparentes sont utiles lorsque vous souhaitez créer des applications aux bordures irrégulières, qui « s'atténuent » ou semblent invisibles. Le rendu des zones de grande taille sur lesquelles le fondu alpha a été appliqué peut toutefois être lent ; par conséquent, n'utilisez cet effet que lorsque cela est strictement nécessaire.

Important : *sous Linux, les événements de souris ne passent pas à travers des pixels entièrement transparents. Évitez de créer des fenêtres comprenant de vastes zones entièrement transparentes car vous pourriez bloquer de façon invisible l'accès de l'utilisateur aux autres fenêtres ou aux autres éléments de son bureau. Sous Mac OS X et Windows, les événements de souris passent à travers les pixels entièrement transparents.*

La transparence ne peut pas être utilisée avec des fenêtres disposant d'un chrome système. En outre, le contenu SWF et PDF dans HTML risque de ne pas s'afficher dans des fenêtres transparentes. Pour plus d'informations, voir « Eléments à prendre en compte lors du chargement d'un contenu SWF ou PDF dans une page HTML » à la page 1045.

La propriété statique `NativeWindow.supportsTransparency` indique si la transparence de la fenêtre est disponible. Si la transparence n'est pas prise en charge, l'application utilise un arrière-plan noir. Dans ce cas, les zones transparentes de l'application s'affichent en noir opaque. Il est recommandé de proposer une solution de remplacement au cas où cette propriété serait définie sur `false`. Vous pourriez ainsi montrer une boîte de dialogue d'avertissement à l'utilisateur ou afficher une interface utilisateur rectangulaire et non transparente.

Notez que la transparence est toujours prise en charge par les systèmes d'exploitation Mac et Windows. La prise en charge sur les systèmes d'exploitation Linux requiert un gestionnaire de composition de fenêtres. Cependant, même lorsqu'un tel gestionnaire est disponible, la transparence peut ne pas être disponible du fait des options d'affichage de l'utilisateur ou de sa configuration matérielle.

Transparence dans une fenêtre d'application MXML

Adobe AIR 1.0 et les versions ultérieures

Par défaut, l'arrière-plan d'une fenêtre MXML est opaque, même si vous créez la fenêtre à l'aide de la propriété *transparent*. (Notez l'effet de transparence aux coins de la fenêtre.) Pour présenter un arrière-plan transparent pour la fenêtre, définissez une couleur d'arrière-plan et une valeur alpha dans la feuille de style ou l'élément <mx:Style> contenu dans le fichier MXML de votre application. Par exemple, la déclaration de style suivante confère à l'arrière-plan une nuance de vert légèrement transparente :

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

Transparence dans une fenêtre d'application HTML

Adobe AIR 1.0 et les versions ultérieures

Par défaut, l'arrière-plan du contenu HTML affiché dans les fenêtres HTML et les objets HTMLLoader est opaque, même si la fenêtre conteneur est transparente. Pour désactiver l'arrière-plan par défaut du contenu HTML, définissez la propriété `paintsDefaultBackground` sur `false`. L'exemple suivant crée un objet HTMLLoader et désactive l'arrière-plan par défaut :

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

Cet exemple utilise JavaScript pour désactiver le chrome par défaut d'une fenêtre HTML :

```
window.htmlLoader.paintsDefaultBackground = false;
```

Si un élément du document HTML définit une couleur d'arrière-plan, l'arrière-plan de cet élément n'est pas transparent. Il est impossible de définir une transparence (ou une opacité) partielle. Vous pouvez néanmoins utiliser une image transparente au format PNG comme arrière-plan d'une page ou d'un élément de page pour obtenir un effet visuel similaire.

Appartenance des fenêtres

Une fenêtre peut être *propriétaire* d'une ou de plusieurs fenêtres. Celles-ci s'affichent toujours devant la fenêtre maître, sont réduites en icône et restaurées conjointement avec elle et sont fermées lors de la fermeture de cette dernière. Il est impossible de transférer ou supprimer l'appartenance des fenêtres. Une fenêtre ne peut appartenir qu'à une seule fenêtre maître, mais peut posséder un nombre illimité d'autres fenêtres.

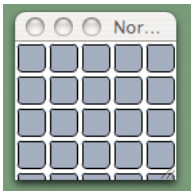
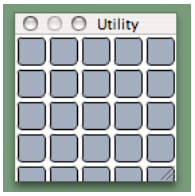
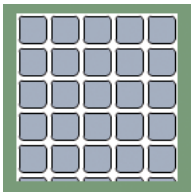
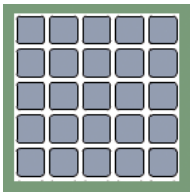
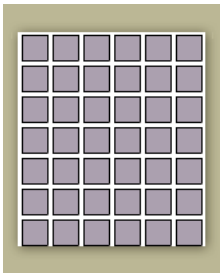
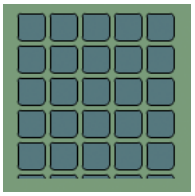
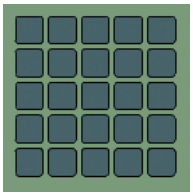
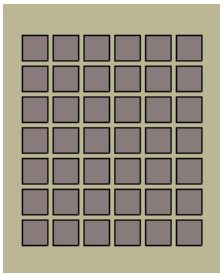
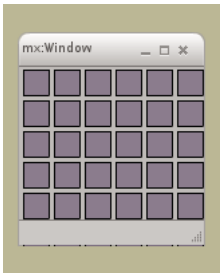
L'appartenance des fenêtres permet de simplifier la gestion des fenêtres servant de palettes d'outils ou de boîtes de dialogue. Supposons, par exemple, que vous affichiez une boîte de dialogue d'enregistrement en conjonction avec une fenêtre de document. Si la boîte de dialogue appartient à la fenêtre de document, elle s'affiche automatiquement devant la fenêtre de document.

- NativeWindow.owner
- Christian Cantrel: Owned windows in AIR 2.6 (disponible en anglais uniquement)

Catalogue des effets visuels d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Le tableau suivant illustre les effets visuels résultant de diverses combinaisons de paramètres de propriétés de fenêtre sous Mac OS X, Windows et Linux :

Paramètres de fenêtre	Mac OS X	Microsoft Windows	Linux*
Type : normale SystemChrome : standard Transparent : false			
Type : utilitaire SystemChrome : standard Transparent : false			
Type : tout type SystemChrome : aucun Transparent : false			
Type : tout type SystemChrome : aucun Transparent : true			
mxWindowedApplication ou mx:Window Type : tout type SystemChrome : aucun Transparent : true			

Ubuntu avec gestionnaire de fenêtres Compiz

Remarque : *les éléments du chrome système suivants ne sont pas pris en charge par AIR : barre d'outils Mac OS X, icône proxy Mac OS X, icônes de la barre de titre Windows et chrome système alternatif.*

Création de fenêtres

Adobe AIR 1.0 et les versions ultérieures

AIR crée automatiquement la première fenêtre d'une application, mais vous pouvez créer autant de fenêtres supplémentaires que vous le souhaitez. Pour créer une fenêtre native, utilisez la méthode constructeur NativeWindow.

Pour créer une fenêtre HTML, utilisez la méthode `createRootWindow()` de l'objet HTMLLoader ou appelez la méthode JavaScript `window.open()` depuis un document HTML. La fenêtre créée est un objet NativeWindow dont la liste d'affichage contient un objet HTMLLoader. L'objet HTMLLoader interprète et affiche le contenu HTML et JavaScript associé à la fenêtre. Vous pouvez accéder aux propriétés de l'objet NativeWindow sous-jacent à partir de JavaScript par le biais de la propriété `window.nativeWindow`. (Cette propriété est réservée au code qui s'exécute dans le sandbox d'application AIR.)

Lorsque vous initialisez une fenêtre, y compris la fenêtre d'application initiale, il peut s'avérer utile de la créer dans l'état invisible, de charger un contenu ou d'exécuter toute mise à jour graphique, puis d'activer la visibilité de la fenêtre. Cette séquence permet d'éviter que l'utilisateur visualise toute modification visuelle peu judicieuse. Pour stipuler que la fenêtre initiale de l'application doit être créée dans l'état invisible, spécifiez la balise `<visible>false</visible>` dans le descripteur d'application (ou omettez complètement la balise, puisque `false` est la valeur par défaut). Les nouvelles fenêtres NativeWindows sont invisibles par défaut. Lorsque vous créez une fenêtre HTML par le biais de la méthode `createRootWindow()` de la classe HTMLLoader, vous pouvez définir l'argument `visible` sur `false`. Appelez la méthode `activate()` de NativeWindowmethod ou définissez la propriété `visible` sur `true` pour activer la visibilité d'une fenêtre.

Spécification des propriétés d'initialisation d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Une fois la fenêtre du bureau créée, il est impossible de modifier les propriétés d'initialisation d'une fenêtre native. Ces propriétés immuables et leurs valeurs par défaut sont les suivantes :

Propriété	Valeur par défaut
systemChrome	standard
type	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

Définissez les propriétés de la fenêtre initiale créée par AIR dans le fichier descripteur d'application. La fenêtre principale d'une application AIR est toujours de type *normale*. (Vous pouvez spécifier d'autres propriétés de fenêtre dans le fichier descripteur, telles que `visible`, `width`, et `height`, et les modifier à tout moment.)

Définissez les propriétés des autres fenêtres natives et HTML créées par votre application à l'aide de la classe NativeWindowInitOptions. Lorsque vous créez une fenêtre, vous devez transmettre un objet NativeWindowInitOptions spécifiant les propriétés de la fenêtre à la fonction constructeur NativeWindow ou à la méthode `createRootWindow()` de l'objet HTMLLoader.

Le code suivant crée un objet NativeWindowInitOptions pour une fenêtre d'utilitaire :

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

Il est impossible de définir *systemChrome* sur *standard* lorsque la propriété *transparent* est définie sur `true` ou lorsque la propriété `type` est définie sur `lightweight`.

Remarque : *vous ne pouvez pas définir les propriétés d'initialisation d'une fenêtre créée avec la fonction `window.open()` de JavaScript. Vous pouvez toutefois passer outre la méthode de création de ces fenêtres en implémentant votre propre classe HTMLHost. Pour plus d'informations, voir la section « Traitement des appels JavaScript à window.open() » à la page 1057.*

Lorsque vous créez une fenêtre à l'aide de la classe mx:Window de Flex, définissez les propriétés d'initialisation sur l'objet window en tant que tel, soit dans la déclaration MXML de la fenêtre, soit dans le code qui crée la fenêtre. L'objet NativeWindow sous-jacent n'est pas créé tant que vous n'avez pas appelé la méthode `open()`. Après avoir ouvert une fenêtre, il est impossible de modifier ces propriétés d'initialisation.

Création de la fenêtre initiale de l'application

Adobe AIR 1.0 et les versions ultérieures

AIR crée la fenêtre initiale de l'application en fonction des propriétés spécifiées dans le fichier descripteur d'application et charge le fichier référencé dans l'élément de contenu. L'élément de contenu doit faire référence à un fichier SWF ou HTML.

La fenêtre initiale peut être la fenêtre principale de votre application, ou permettre simplement de lancer une ou plusieurs autres fenêtres. Il n'est absolument pas nécessaire de rendre cette fenêtre visible.

Création de la fenêtre initiale à l'aide d'ActionScript

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez une application AIR à l'aide d'ActionScript, la classe principale de votre application doit étendre la classe Sprite (ou une sous-classe de cette dernière). Cette classe sert de point d'entrée principal à l'application.

Lorsque votre application s'ouvre, AIR crée une fenêtre, crée une occurrence de la classe principale, puis ajoute l'occurrence à la scène de la fenêtre. Pour accéder à la fenêtre, vous pouvez écouter l'événement `addedToStage`, puis utiliser la propriété `nativeWindow` de l'objet Stage pour obtenir une référence à l'objet NativeWindow.

L'exemple suivant illustre le squelette de base de la classe principale d'une application AIR créée avec ActionScript :

```
package {
    import flash.display.NativeWindow;
    import flash.display.Sprite;
    import flash.events.Event;

    public class MainClass extends Sprite
    {
        private var mainWindow:NativeWindow;
        public function MainClass(){
            this.addEventListener(Event.ADDED_TO_STAGE, initialize);
        }

        private function initialize(event:Event):void{
            mainWindow = this.stage.nativeWindow;
            //perform initialization...
            mainWindow.activate(); //show the window
        }
    }
}
```

Remarque : *techniquement, vous POUVEZ accéder à la propriété* `nativeWindow` *de la fonction constructeur de la classe principale. Il s'agit cependant d'un cas spécial qui ne s'applique qu'à la fenêtre d'application initiale.*

Lorsque vous créez une application dans Flash Professional, la classe de document principale est automatiquement générée si vous n'en créez pas une dans un fichier ActionScript distinct. Vous pouvez accéder à l'objet NativeWindow associé à la fenêtre initiale par le biais de la propriété `nativeWindow` de la scène. A titre d'exemple, le code suivant active la fenêtre principale dans un état d'affichage maximal (à partir du scénario) :

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

Création de la fenêtre initiale à l'aide de Flex

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous créez une application AIR à l'aide de la structure d'application Flex, utilisez mx:WindowedApplication comme élément racine de votre fichier MXML principal. (Vous pouvez utiliser le composant mx:Application, mais il ne prend pas en charge toutes les fonctionnalités intégrées à AIR.) Le composant WindowedApplication sert de point d'entrée initial à l'application.

Lorsque vous lancez l'application, AIR crée une fenêtre native, initialise la structure d'application Flex et ajoute l'objet WindowedApplication à la scène de la fenêtre. Une fois la séquence de lancement terminée, l'objet WindowedApplication distribue un événement `applicationComplete`. Accédez à l'objet de la fenêtre du bureau avec la propriété `nativeWindow` de l'occurrence de WindowedApplication.

L'exemple suivant crée un composant WindowedApplication simple qui définit ses coordonnées x et y :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    applicationComplete="placeWindow()">
    <mx:Script>
        <![CDATA[
            private function placeWindow():void{
                this.nativeWindow.x = 300;
                this.nativeWindow.y = 300;
            }
        ]]>
    </mx:Script>
    <mx:Label text="Hello World" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

Création d'une classe NativeWindow

Adobe AIR 1.0 et les versions ultérieures

Pour créer une classe NativeWindow, transmettez un objet NativeWindowInitOptions au constructeur NativeWindow :

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

La fenêtre ne s'affiche pas tant que vous ne définissez pas la propriété `visible` sur `true` ou tant que vous n'appelez pas la méthode `activate()`.

Une fois la fenêtre créée, vous pouvez initialiser ses propriétés et charger le contenu dans la fenêtre à l'aide de la propriété stage et des techniques de la liste d'affichage de Flash.

Dans la plupart des cas, vous devez définir la propriété `scaleMode` de la scène d'une nouvelle fenêtre native sur `noScale` (utilisez la constante `StageScaleMode.NO_SCALE`). Les modes d'échelle de Flash sont conçus pour des situations dans lesquelles l'auteur de l'application ne connaît pas à l'avance le format de l'espace d'affichage de l'application. Les modes d'échelle permettent à l'auteur de faire le moins mauvais choix : découper le contenu, l'étirer ou le compresser, ou le remplir d'espace vide. Etant donné que vous contrôlez l'espace d'affichage dans AIR (le cadre de la fenêtre), vous pouvez dimensionner la fenêtre en fonction du contenu ou dimensionner le contenu en fonction de la fenêtre sans aucun compromis.

Le mode d'échelle pour Flex et les fenêtres HTML est automatiquement défini sur `noScale`.

Remarque : *pour déterminer les tailles maximale et minimale des fenêtres disponibles sur le système d'exploitation, utilisez les propriétés NativeWindow statiques suivantes :*

```
var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

Création d'une fenêtre HTML

Adobe AIR 1.0 et les versions ultérieures

Pour créer une fenêtre HTML, vous pouvez appeler la méthode `Window.open()` de JavaScript ou appeler la méthode `createRootWindow()` de la classe HTMLLoader d'AIR.

Le contenu HTML dans les sandbox de sécurité peuvent utiliser la méthode standard `Window.open()` de JavaScript. Si le contenu est exécuté hors du sandbox de l'application, la méthode `open()` ne peut être appelée qu'en réponse à l'interaction de l'utilisateur, notamment lorsque ce dernier clique sur la souris ou appuie sur une touche. Lorsque la méthode `open()` est appelée, une fenêtre avec un chrome système est créée pour afficher le contenu à l'URL spécifiée. Exemple :

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

Remarque : *vous pouvez étendre la classe HTMLHost dans ActionScript pour personnaliser la fenêtre créée avec la fonctionnalité `window.open()` de JavaScript. Voir la section « A propos de l'extension de la classe HTMLHost » à la page 1049.*

Le contenu du sandbox de sécurité de l'application a accès à la méthode de création de fenêtres la plus performante : `HTMLLoader.createRootWindow()`. Cette méthode vous permet de spécifier toutes les options de création d'une nouvelle fenêtre. Par exemple, le code JavaScript suivant crée un type de fenêtre légère sans chrome système, d'une taille de 300x400 pixels :

```
 var options = new air.NativeWindowInitOptions();
options.systemChrome = "none";
options.type = "lightweight";

var windowBounds = new air.Rectangle(200,250,300,400);
newHTMLLoader = air.HTMLLoader.createRootWindow(true, options, true, windowBounds);
newHTMLLoader.load(new air.URLRequest("xmpl.html"));
```

Remarque : *si le contenu chargé par la nouvelle fenêtre se trouve hors du sandbox de sécurité de l'application, l'objet window ne possède aucune propriété AIR : `runtime`, `nativeWindow` ou `htmlLoader`.*

Si vous créez une fenêtre transparente, le contenu SWF intégré au code HTML chargé dans cette fenêtre ne s'affiche pas systématiquement. Vous devez définir le paramètre `wmode` de l'objet ou de la balise embed qui fait référence au fichier SWF sur `opaque` ou `transparent`. Etant donné que la valeur par défaut de `wmode` est `window`, le contenu SWF ne s'affiche par défaut pas dans une fenêtre transparente. Un contenu PDF ne s'affiche pas dans une fenêtre transparente, quelle que soit la valeur définie de `wmode`. (Dans les versions d'AIR antérieures à 1.5.2, il était également d'impossible d'afficher un contenu SWF dans une fenêtre transparente.)

Les fenêtres créées avec la méthode `createRootWindow()` restent indépendantes de la fenêtre d'ouverture. Les propriétés `parent` et `opener` de l'objet Window de JavaScript sont définies sur `null`. La fenêtre d'ouverture peut accéder à l'objet Window de la nouvelle fenêtre à l'aide de la référence HTMLLoader renvoyée par la fonctionnalité `createRootWindow()`. Dans le cadre de l'exemple précédent, l'instruction `newHTMLLoader.window` référencerait l'objet Window de JavaScript de la fenêtre créée.

Remarque : *la fonctionnalité `createRootWindow()` peut être appelée depuis JavaScript et ActionScript.*

Création d'un composant mx:Window

Adobe AIR 1.0 et les versions ultérieures

Pour créer un composant mx:Window, vous pouvez créer un fichier MXML à l'aide de mx:Window comme balise racine ou appeler directement le constructeur de la classe Window.

L'exemple suivant crée et affiche un composant mx:Window en appelant le constructeur Window :

```
 var newWindow:Window = new Window();
newWindow.systemChrome = NativeWindowSystemChrome.NONE;
newWindow.transparent = true;
newWindow.title = "New Window";
newWindow.width = 200;
newWindow.height = 200;
newWindow.open(true);
```

Ajout de contenu à une fenêtre

Adobe AIR 1.0 et les versions ultérieures

La méthode d'ajout de contenu à une fenêtre AIR dépend du type de fenêtre. Ainsi, MXML et HTML vous permettent de définir d'une manière déclarative le contenu de base de la fenêtre. Vous pouvez intégrer des ressources dans les fichiers SWF de l'application ou les charger à partir de fichiers d'application distincts. Le contenu Flex, Flash et HTML peut être créé au moment opportun et ajouté à la fenêtre de façon dynamique.

Lorsque vous chargez du contenu SWF ou du contenu HTML contenant JavaScript, vous devez tenir compte du modèle de sécurité d'AIR. Le contenu dans le sandbox de sécurité de l'application, c'est-à-dire le contenu installé avec votre application et pouvant être chargé avec le modèle d'URL app:, dispose de privilèges complets pour accéder à toutes les API d'AIR. Le contenu chargé hors de ce sandbox ne peut pas accéder aux API d'AIR. Le contenu JavaScript hors du sandbox de l'application ne peut pas utiliser les propriétés `runtime`, `nativeWindow` et `htmlLoader` de l'objet Window de JavaScript.

Pour sécuriser la programmation croisée, vous pouvez utiliser un pont de sandbox afin de fournir une interface limitée entre le contenu applicatif et le contenu hors application. Dans le contenu HTML, vous pouvez également mapper les pages de votre application sur un sandbox hors application afin que le code sur cette page puisse intercoder le contenu externe. Voir le chapitre « Sécurité AIR » à la page 1122.

Chargement d'une image ou d'un fichier SWF

Vous pouvez charger des images ou des fichiers Flash SWF dans la liste d'affichage d'une fenêtre native à l'aide de la classe `flash.display.Loader` :

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.display.Loader;

    public class LoadedSWF extends Sprite
    {
        public function LoadedSWF(){
            var loader:Loader = new Loader();
            loader.load(new URLRequest("visual.swf"));
            loader.contentLoaderInfo.addEventListener(Event.COMPLETE,loadFlash);
        }

        private function loadFlash(event:Event):void{
            addChild(event.target.loader);
        }
    }
}
```

Remarque : *les anciens fichiers SWF créés à l'aide d'ActionScript 1 ou 2 partagent les états globaux, tels que les définitions de classe, les singletons et les variables globales, si ceux-ci sont chargés dans la même fenêtre. Si ces fichiers SWF dépendent d'états globaux intacts pour fonctionner correctement, ils ne peuvent pas être chargés plus d'une fois dans la même fenêtre ; ils ne peuvent en outre pas être chargés dans la même fenêtre en tant qu'autre fichier SWF utilisant des définitions de classe et des variables qui se chevauchent. Ce contenu peut être chargé dans des fenêtres indépendantes.*

Chargement du contenu HTML dans une fenêtre native

Pour charger du contenu HTML dans une fenêtre native, vous pouvez ajouter un objet HTMLLoader à la scène de la fenêtre, puis charger le contenu HTML dans cet objet, ou créer une fenêtre qui contient déjà un objet HTMLLoader à l'aide de la méthode `HTMLLoader.createRootWindow()`. L'exemple suivant affiche un contenu HTML dans une zone d'affichage de 300x500 pixels sur la scène d'une fenêtre native :

```
 //newWindow is a NativeWindow instance
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.width = 300;
htmlView.height = 500;

//set the stage so display objects are added to the top-left and not scaled
newWindow.stage.align = "TL";
newWindow.stage.scaleMode = "noScale";
newWindow.stage.addChild( htmlView );

//urlString is the URL of the HTML page to load
htmlView.load( new URLRequest(urlString) );
```

Pour charger une page HTML dans une application Flex, vous pouvez utiliser le composant HTML Flex.

Un contenu SWF intégré à un fichier HTML ne s'affiche pas si la fenêtre est transparente (en d'autres termes, si la propriété `transparent` de la fenêtre est définie sur `true`), sauf si le paramètre `wmode` de l'objet ou de la balise embed qui fait référence au fichier SWF est défini sur `opaque` ou `transparent`. Etant donné que la valeur par défaut de `wmode` est `window`, un contenu SWF ne s'affiche par défaut pas dans une fenêtre transparente. Un contenu PDF ne s'affiche pas dans une fenêtre transparente, quelle que soit la valeur `wmode` utilisée.

Par ailleurs, un contenu SWF ou PDF ne s'affiche pas si le contrôle HTMLLoader est mis à l'échelle ou pivoté ou si sa propriété `alpha` est définie sur une valeur autre que 1.0.

Incrustation du contenu SWF sur une fenêtre HTML

Etant donné que les fenêtres HTML sont contenues dans une occurrence de NativeWindow, vous pouvez ajouter des objets d'affichage Flash au-dessus et en dessous du calque HTML dans la liste d'affichage.

Pour ajouter un objet d'affichage au-dessus du calque HTML, utilisez la méthode `addChild()` de la propriété `window.nativeWindow.stage`. La méthode `addChild()` ajoute le contenu superposé au-dessus du contenu existant dans la fenêtre.

Pour ajouter un objet d'affichage en dessous du calque HTML, utilisez la méthode `addChildAt()` de la propriété `window.nativeWindow.stage`, en transmettant une valeur de zéro au paramètre `index`. Le fait de placer un objet à l'index zéro déplace le contenu existant (notamment l'affichage HTML) au-dessus d'un calque et insère le nouveau contenu en dessous. Afin que le contenu superposé sous la page HTML soit visible, vous devez définir la propriété `paintsDefaultBackground` de l'objet `HTMLlLoader` sur `false`. En outre, tout élément de la page qui définit une couleur d'arrière-plan ne sera pas transparent. Par exemple, si vous définissez une couleur d'arrière-plan pour l'élément de corps de la page, aucune page ne sera transparente.

L'exemple suivant explique comment ajouter des objets d'affichage Flash en tant que sur-couches et sous-couches à une page HTML. L'exemple suivant crée deux objets de formes simples, ajoute le premier en dessous du contenu HTML et l'autre au-dessus. Cet exemple met également à jour la position de la forme en fonction de l'événement `enterFrame`.

```
 <html>
<head>
<title>Bouncers</title>
<script src="AIRAliases.js" type="text/javascript"></script>
<script language="JavaScript" type="text/javascript">
air.Shape = window.runtime.flash.display.Shape;

function Bouncer(radius, color){
    this.radius = radius;
    this.color = color;

    //velocity
    this.vX = -1.3;
    this.vY = -1;

    //Create a Shape object and draw a circle with its graphics property
    this.shape = new air.Shape();
    this.shape.graphics.lineStyle(1,0);
    this.shape.graphics.beginFill(this.color,.9);
    this.shape.graphics.drawCircle(0,0,this.radius);
    this.shape.graphics.endFill();

    //Set the starting position
    this.shape.x = 100;
    this.shape.y = 100;


    //Moves the sprite by adding (vX,vY) to the current position
    this.update = function(){
        this.shape.x += this.vX;
        this.shape.y += this.vY;

        //Keep the sprite within the window
        if( this.shape.x - this.radius < 0){
            this.vX = -this.vX;
        }
        if( this.shape.y - this.radius < 0){
            this.vY = -this.vY;
        }
        if( this.shape.x  + this.radius > window.nativeWindow.stage.stageWidth){
            this.vX = -this.vX;
        }
        if( this.shape.y  + this.radius > window.nativeWindow.stage.stageHeight){
            this.vY = -this.vY;
        }

    };
}

function init(){
    //turn off the default HTML background
```

```
    window.htmlLoader.paintsDefaultBackground = false;
    var bottom = new Bouncer(60,0xff2233);
    var top = new Bouncer(30,0x2441ff);

    //listen for the enterFrame event
    window.htmlLoader.addEventListener("enterFrame",function(evt){
        bottom.update();
        top.update();
    });

    //add the bouncing shapes to the window stage
    window.nativeWindow.stage.addChildAt(bottom.shape,0);
    window.nativeWindow.stage.addChild(top.shape);
}
</script>
<body onload="init();">
<h1>de Finibus Bonorum et Malorum</h1>
<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis
et quasi architecto beatae vitae dicta sunt explicabo.</p>
<p style="background-color:#FFFF00; color:#660000;">This paragraph has a background color.</p>
<p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis
praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias
excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui
officia deserunt mollitia animi, id est laborum et dolorum fuga.</p>
</body>
</html>
```

Cet exemple vous donne des notions rudimentaires sur certaines techniques avancées qui utilisent JavaScript et ActionScript dans AIR. Si vous ne maîtrisez pas l'utilisation des objets d'affichage ActionScript, voir la section « Programmation de l'affichage » à la page 156 du *Guide du développeur d'ActionScript 3.0*.

Exemple : Création d'une fenêtre native

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant illustre la méthode de création d'une fenêtre native :

```
public function createNativeWindow():void {
    //create the init options
    var options:NativeWindowInitOptions = new NativeWindowInitOptions();
    options.transparent = false;
    options.systemChrome = NativeWindowSystemChrome.STANDARD;
    options.type = NativeWindowType.NORMAL;

    //create the window
    var newWindow:NativeWindow = new NativeWindow(options);
    newWindow.title = "A title";
    newWindow.width = 600;
    newWindow.height = 400;

    newWindow.stage.align = StageAlign.TOP_LEFT;
    newWindow.stage.scaleMode = StageScaleMode.NO_SCALE;

    //activate and show the new window
    newWindow.activate();
}
```

Gestion des fenêtres

Adobe AIR 1.0 et les versions ultérieures

Utilisez les propriétés et les méthodes de la classe NativeWindow pour gérer l'aspect, le comportement et le cycle de vie des fenêtres du poste de travail.

Remarque : *si vous faites appel à la structure Flex, il est généralement préférable de gérer le comportement d'une fenêtre à l'aide des classes framework. Vous pouvez accéder à la plupart des propriétés et méthodes NativeWindow par le biais des classes mx:WindowedApplication et mx:Window.*

Obtention d'une occurrence de NativeWindow

Adobe AIR 1.0 et les versions ultérieures

Pour manipuler une fenêtre, vous devez tout d'abord obtenir l'occurrence de la fenêtre. Vous pouvez obtenir une occurrence de fenêtre à partir de l'un des emplacements suivants :

- Le constructeur de fenêtre natif utilisé pour créer la fenêtre :

  ```
  var win:NativeWindow = new NativeWindow(initOptions);
  ```

- La propriété `nativeWindow` de la scène de la fenêtre :

  ```
  var win:NativeWindow = stage.nativeWindow;
  ```

- La propriété `stage` d'un objet d'affichage de la fenêtre :

  ```
  var win:NativeWindow = displayObject.stage.nativeWindow;
  ```

- La propriété `target` d'un événement de fenêtre native distribué par la fenêtre :

  ```
  private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
  {
      var win:NativeWindow = event.target as NativeWindow;
  }
  ```

- La propriété `nativeWindow` d'une page HTML affichée dans la fenêtre :

  ```
  var win:NativeWindow = htmlLoader.window.nativeWindow;
  ```

- Les propriétés `activeWindow` et `openedWindows` de l'objet NativeApplication :

  ```
  var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
  var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
  ```

  `NativeApplication.nativeApplication.activeWindow` référence la fenêtre active d'une application (mais renvoie `null` si la fenêtre active n'est pas une fenêtre de cette application AIR). Le tableau `NativeApplication.nativeApplication.openedWindows` contient toutes les fenêtres d'une application AIR n'ayant pas été fermée.

Les objets Flex mx:WindowedApplication et mx:Window étant des objets d'affichage, vous pouvez aisément référencer la fenêtre d'application dans un fichier MXML à l'aide de la propriété `stage`, comme suit :

```
 <?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
applicationComplete="init();">
    <mx:Script>
        <![CDATA[
            import flash.display.NativeWindow;

            public function init():void{
                var appWindow:NativeWindow = this.stage.nativeWindow;
                //set window properties
                appWindow.visible = true;
            }
        ]]>
    </mx:Script>
</WindowedApplication
```

Remarque : *tant que le composant WindowedApplication ou Window n'est pas ajouté à la scène de la fenêtre à l'aide de la structure d'application Flex, la propriété `stage` du composant est `null`. Ce comportement correspond à celui du composant Flex Application ; cela signifie qu'il n'est pas possible d'accéder à la scène ou à l'occurrence de NativeWindow dans les écouteurs pour les événements qui se produisent avec antériorité dans le cycle d'initialisation des composants WindowedApplication et Window, tels que `creationComplete`. Il est recommandé d'accéder à la scène et à l'occurrence de NativeWindow lorsque l'événement `applicationComplete` est distribué.*

Activation, affichage et masquage des fenêtres

Adobe AIR 1.0 et les versions ultérieures

Pour activer une fenêtre, appelez la méthode NativeWindow `activate()`. Lorsque vous activez une fenêtre, celle-ci s'affiche au premier plan, la souris et le clavier reçoivent le focus et, le cas échéant, il est possible de rendre visible la fenêtre en la restaurant ou en définissant la propriété `visible` sur `true`. L'activation d'une fenêtre ne modifie pas l'ordre des autres fenêtres dans l'application. Lorsque vous appelez la méthode `activate()`, la fenêtre distribue un événement `activate`.

Pour afficher une fenêtre masquée sans l'activer, définissez la propriété `visible` sur `true`. Cette action amène la fenêtre au premier plan, mais ne lui donne pas le focus.

Pour masquer une fenêtre, définissez sa propriété `visible` sur `false`. Le masquage d'une fenêtre supprime l'affichage de la fenêtre et des icônes de la barre des tâches associées et, sous Mac OS X, l'entrée dans le menu Fenêtre.

Lorsque vous modifiez la visibilité d'une fenêtre, la visibilité des fenêtres dont elle est propriétaire est également modifiée. Ainsi, si vous masquez une fenêtre, toutes les fenêtres qui lui appartiennent sont également masquées.

Remarque : *sous Mac OS X, il n'est pas possible de masquer entièrement une fenêtre réduite qui possède une icône dans la partie de la fenêtre du Dock. Si la propriété* `visible` *est définie sur* `false` *sur une fenêtre minimisée, l'icône de la fenêtre dans le Dock est toujours affichée. Si l'utilisateur clique sur l'icône, la fenêtre est restaurée à un état visible et s'affiche.*

Modification de l'ordre d'affichage des fenêtres

Adobe AIR 1.0 et les versions ultérieures

AIR propose plusieurs méthodes pour modifier directement l'ordre d'affichage des fenêtres. Vous pouvez placer une fenêtre au premier plan ou à l'arrière-plan ; vous pouvez en outre placer une fenêtre sur ou sous une autre fenêtre. L'utilisateur peut simultanément réorganiser les fenêtres en les activant.

Il est possible de maintenir une fenêtre au premier plan en définissant sa propriété `alwaysInFront` sur `true`. Si ce même paramètre est défini dans plusieurs fenêtres, l'ordre d'affichage est à nouveau déterminé au sein des ces fenêtres ; toutefois, ces fenêtres sont toujours placées au-dessus des fenêtres dont la propriété `alwaysInFront` est définie sur false.

Les fenêtres du groupe de premier niveau sont toujours affichées devant les fenêtres d'autres applications, même si l'application AIR n'est pas active. Etant donné que ce comportement peut perturber les utilisateurs, définissez la propriété `alwaysInFront` sur `true` uniquement lorsque cela est nécessaire. Exemples de cas dans lesquels cet emploi est justifié :

- Fenêtres contextuelles temporaires, notamment infos-bulles, listes déroulantes, menus personnalisés ou zones déroulantes. Ces fenêtres doivent se fermer lorsqu'elles perdent le focus ; or, il est possible d'éviter qu'un utilisateur ne puisse pas afficher une autre fenêtre.
- Messages et alertes extrêmement urgents. Lorsqu'une modification irrévocable risque de se produire si l'utilisateur ne répond pas à temps, il peut être justifié d'afficher au premier plan une fenêtre d'alerte. Néanmoins, la plupart des erreurs et des alertes peuvent être gérées dans l'ordre d'affichage normal des fenêtres.
- Fenêtres affichées temporairement l'une derrière l'autre.

Remarque : *AIR n'impose pas l'utilisation correcte de la propriété* `alwaysInFront`*. Néanmoins, si votre application interrompt le flux de travail d'un utilisateur, il est probable qu'elle termine dans la corbeille de cet utilisateur.*

Si une fenêtre est propriétaire d'autres fenêtres, celles-ci s'affichent devant elle dans un ordre déterminé. Si vous appelez `orderToFront()` ou définissez `alwaysInFront` sur `true` pour une fenêtre propriétaire d'autres fenêtres, l'ordre d'affichage de ces dernières et celui de la fenêtre propriétaire est modifié devant d'autres fenêtres, mais elles continuent à s'afficher devant la fenêtre propriétaire.

Appeler les méthodes de classement pour des fenêtres qui appartiennent à une autre fenêtre fonctionne normalement si elles appartiennent toutes à un même propriétaire, mais risque également de modifier l'ordre d'affichage du groupe entier de fenêtres. Ainsi, si vous appelez `orderToFront()` pour une fenêtre appartenant à une autre fenêtre, la fenêtre, son propriétaire et toute fenêtre appartenant éventuellement au même propriétaire sont affichées devant les autres fenêtres.

La classe NativeWindow fournit les propriétés et les méthodes suivantes pour définir l'ordre d'affichage d'une fenêtre par rapport à d'autres fenêtres :

Membre	Description
alwaysInFront, propriété	Indique si la fenêtre est affichée dans le groupe de fenêtres de premier niveau. Dans la plupart des cas, le paramètre `false` est recommandé. Si vous modifiez la valeur de `false` à `true`, la fenêtre est placée devant toutes les autres fenêtres (mais elle n'est pas activée). Si vous modifiez la valeur de `true` à `false`, la fenêtre est placée derrière les fenêtres restantes du groupe de premier niveau, mais reste devant les autres fenêtres. Si vous définissez la propriété sur sa valeur actuelle pour une fenêtre, vous ne modifiez pas l'ordre d'affichage de cette dernière. Le paramètre `alwaysInFront` n'affecte pas les fenêtres qui appartiennent à une autre fenêtre.
orderToFront()	Place la fenêtre au premier plan.
orderInFrontOf()	Place la fenêtre directement devant une fenêtre spécifique.
orderToBack()	Envoie la fenêtre derrière les autres fenêtres.
orderBehind()	Envoie la fenêtre directement derrière une fenêtre spécifique.
activate()	Place la fenêtre au premier plan (tout en la rendant visible et en lui donnant le focus).

Remarque : *si une fenêtre est masquée (la propriété `visible` est définie sur `false`) ou minimisée, l'appel des méthodes de l'ordre d'affichage est sans effet.*

Dans le système d'exploitation Linux, les différents gestionnaires de fenêtres imposent des règles différentes quant à l'ordre d'affichage des fenêtres :

- Dans certains gestionnaires de fenêtres, les fenêtres d'utilitaires sont toujours affichées devant les fenêtres normales.
- Dans certains gestionnaires de fenêtres, une fenêtre plein écran dont la propriété `alwaysInFront` est définie sur `true` s'affiche toujours devant les autres fenêtres dont la propriété `alwaysInFront` est également définie sur `true`.

Fermeture d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Pour fermer une fenêtre, utilisez la méthode `NativeWindow.close()`.

La fermeture d'une fenêtre décharge le contenu de la fenêtre, mais si les autres objets possèdent des références à ce contenu, les objets de contenu ne sont pas détruits. La méthode `NativeWindow.close()` s'exécute de façon asynchrone, et l'application contenue dans la fenêtre continue de fonctionner lors de la fermeture. La méthode de fermeture distribue un événement close lorsque l'opération de fermeture est terminée. Techniquement, l'objet NativeWindow est toujours valide, mais l'accès à la plupart des propriétés et des méthodes sur une fenêtre fermée génère un erreur IllegalOperationError. Vous ne pouvez pas ouvrir à nouveau une fenêtre fermée. Vérifiez la propriété `closed` d'une fenêtre pour vous assurer que la fenêtre a été fermée. Pour simplement masquer une fenêtre, définissez la propriété `NativeWindow` sur `false`.

Si la propriété `Nativeapplication.autoExit` est définie sur `true` (valeur par défaut), l'application se ferme lors de la fermeture de sa dernière fenêtre.

La fermeture d'une fenêtre qui possède d'autres fenêtres entraîne la fermeture de ces dernières. Etant donné que les fenêtres qui appartiennent à la fenêtre propriétaire ne distribuent pas d'événement closing, il est impossible d'interdire leur fermeture. Un événement close est distribué.

Autorisation d'annulation des opérations sur les fenêtres

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'une fenêtre utilise le chrome système, l'interaction de l'utilisateur avec la fenêtre peut être annulée en écoutant les événements appropriés et en annulant leur comportement par défaut. Par exemple, lorsqu'un utilisateur clique sur le bouton de fermeture du chrome système, l'événement `closing` est distribué. Si l'un des écouteurs enregistrés appelle la méthode `preventDefault()` de l'événement, la fenêtre ne se ferme pas.

Si une fenêtre n'utilise pas le chrome système, les événements de notification des modifications désirées ne sont pas automatiquement distribués avant que la modification ait lieu. Donc, si vous appelez les méthodes qui permettent de fermer une fenêtre, d'en modifier l'état ou d'en définir les propriétés de limite, la modification ne peut pas être annulée. Pour notifier les composants dans votre application avant d'effectuer une modification, votre logique d'application peut distribuer l'événement de notification adéquat à l'aide de la méthode `dispatchEvent()` de la fenêtre.

Par exemple, la logique suivante implémente un gestionnaire d'événement pouvant être annulé pour le bouton de fermeture d'une fenêtre :

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

La méthode `dispatchEvent()` renvoie `false` si la méthode `preventDefault()` de l'événement est appelée par un écouteur. Toutefois, elle peut également renvoyer `false` pour d'autres raisons ; par conséquent, il convient d'utiliser explicitement la méthode `isDefaultPrevented()` afin de vérifier si la modification doit ou non être annulée.

Agrandissement, réduction et restauration d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Pour agrandir la fenêtre, utilisez la méthode `maximize()` de la classe NativeWindow.

```
myWindow.maximize();
```

Pour réduire la fenêtre, utilisez la méthode `minimize()` de la classe NativeWindow.

```
myWindow.minimize();
```

Pour restaurer la fenêtre (c'est-à-dire rétablir la taille de la fenêtre avant l'agrandissement ou la réduction), utilisez la méthode `restore()` de la classe NativeWindow.

```
myWindow.restore();
```

Une fenêtre appartenant à une autre fenêtre est réduite en icône ou restaurée lorsque la fenêtre propriétaire est réduite en icône ou restaurée. Etant donné que sa propriétaire est réduite en icône, aucun événement n'est distribué par la fenêtre lorsqu'elle est réduite en icône.

Remarque : *le comportement résultant de l'agrandissement d'une fenêtre AIR est différent du comportement standard de Mac OS X. Plutôt que de basculer entre la taille « standard » définie par l'application et la dernière taille définie par l'utilisateur, les fenêtres AIR basculent entre la dernière taille définie par l'application ou l'utilisateur et la totalité de la zone utilisable de l'écran.*

Dans le système d'exploitation Linux, les différents gestionnaires de fenêtres imposent des règles différentes quant à la définition de l'état d'affichage des fenêtres :

- Avec certains gestionnaires de fenêtres, les fenêtres d'utilitaires ne peuvent pas être agrandies.
- Si une taille maximale est définie pour la fenêtre, certains gestionnaires n'autorisent pas l'agrandissement de la fenêtre. D'autres gestionnaires de fenêtres définissent l'état d'affichage sur l'état maximal, mais ne redimensionnent pas la fenêtre. Dans tous les cas, aucun événement de modification d'état d'affichage n'est distribué.
- Certains gestionnaires de fenêtres ne respectent pas les paramètres `maximizable` ou `minimizable` des fenêtres.

Remarque : *sous Linux, les propriétés des fenêtres sont modifiées de façon asynchrone. Si vous modifiez l'état d'affichage sur une ligne de votre programme et que vous lisez la valeur à la ligne suivante, la modification n'est pas répercutée. Sur toutes les plates-formes, l'objet NativeWindow distribue l'événement* `displayStateChange` *lors de la modification de l'état d'affichage. Si vous devez exécuter une action basée sur le nouvel état de la fenêtre, faites systématiquement appel à un gestionnaire d'événement* `displayStateChange`*. Pour plus d'informations, voir « Ecoute des événements d'une fenêtre » à la page 951.*

Exemple : Réduction, agrandissement, restauration et fermeture d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

La brève application MXML suivante illustre les méthodes `maximize()`, `minimize()`, `restore()` et `close()` de la classe Window :

```
<?xml version="1.0" encoding="utf-8"?>

<mx:WindowedApplication
    xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical">


    <mx:Script>
    <![CDATA[
    public function minimizeWindow():void
    {
        this.stage.nativeWindow.minimize();
    }

    public function maximizeWindow():void
    {
        this.stage.nativeWindow.maximize();
    }

    public function restoreWindow():void
    {
        this.stage.nativeWindow.restore();
    }

    public function closeWindow():void
    {
        this.stage.nativeWindow.close();
    }
    ]]>
    </mx:Script>

    <mx:VBox>
        <mx:Button label="Minimize" click="minimizeWindow()"/>
        <mx:Button label="Restore" click="restoreWindow()"/>
        <mx:Button label="Maximize" click="maximizeWindow()"/>
        <mx:Button label="Close" click="closeWindow()"/>
    </mx:VBox>

</mx:WindowedApplication>
```

L'exemple ActionScript suivant pour Flash crée quatre champs de texte cliquables qui déclenchent les méthodes `minimize()`, `maximize()`, `restore()` et `close()` de la classe NativeWindow :

```
package
{
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.text.TextField;

    public class MinimizeExample extends Sprite
    {
        public function MinimizeExample():void
        {
            var minTextBtn:TextField = new TextField();
            minTextBtn.x = 10;
            minTextBtn.y = 10;
            minTextBtn.text = "Minimize";
            minTextBtn.background = true;
            minTextBtn.border = true;
            minTextBtn.selectable = false;
            addChild(minTextBtn);
            minTextBtn.addEventListener(MouseEvent.CLICK, onMinimize);

            var maxTextBtn:TextField = new TextField();
            maxTextBtn.x = 120;
            maxTextBtn.y = 10;
            maxTextBtn.text = "Maximize";
            maxTextBtn.background = true;
            maxTextBtn.border = true;
            maxTextBtn.selectable = false;
            addChild(maxTextBtn);
            maxTextBtn.addEventListener(MouseEvent.CLICK, onMaximize);

            var restoreTextBtn:TextField = new TextField();
            restoreTextBtn.x = 230;
            restoreTextBtn.y = 10;
            restoreTextBtn.text = "Restore";
            restoreTextBtn.background = true;
            restoreTextBtn.border = true;
            restoreTextBtn.selectable = false;
            addChild(restoreTextBtn);
            restoreTextBtn.addEventListener(MouseEvent.CLICK, onRestore);

            var closeTextBtn:TextField = new TextField();
            closeTextBtn.x = 340;
            closeTextBtn.y = 10;
            closeTextBtn.text = "Close Window";
            closeTextBtn.background = true;
            closeTextBtn.border = true;
            closeTextBtn.selectable = false;
            addChild(closeTextBtn);
```

```
            closeTextBtn.addEventListener(MouseEvent.CLICK, onCloseWindow);
        }
        function onMinimize(event:MouseEvent):void
        {
            this.stage.nativeWindow.minimize();
        }
        function onMaximize(event:MouseEvent):void
        {
            this.stage.nativeWindow.maximize();
        }
        function onRestore(event:MouseEvent):void
        {
            this.stage.nativeWindow.restore();
        }
        function onCloseWindow(event:MouseEvent):void
        {
            this.stage.nativeWindow.close();
        }
    }
}
```

Redimensionnement et déplacement d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'une fenêtre utilise le chrome système, le chrome fournit des commandes de glissement pour redimensionner la fenêtre et la déplacer dans le poste de travail. Si une fenêtre n'utilise pas le chrome système, vous devez ajouter vos propres commandes pour permettre à l'utilisateur de redimensionner et de déplacer la fenêtre.

Remarque : *pour redimensionner ou déplacer une fenêtre, vous devez tout d'abord obtenir une référence à l'occurrence de NativeWindow. Pour plus d'informations sur la méthode d'obtention d'une référence de fenêtre, voir la section* « *Obtention d'une occurrence de NativeWindow* » *à la page 940.*

Redimensionnement d'une fenêtre

Pour que l'utilisateur puisse interagir avec la taille d'une fenêtre, utilisez la méthode `startResize()` de la classe NativeWindow. Lorsque cette méthode est appelée depuis un événement `mouseDown`, l'opération de redimensionnement est effectuée par la souris et se termine lorsque le système d'exploitation reçoit un événement `mouseUp`. Lors de l'appel de la méthode `startResize()`, vous transmettez un argument qui indique le bord ou le coin à partir duquel la fenêtre doit être redimensionnée.

Pour définir la taille de la fenêtre par programmation, définissez les propriétés `width`, `height` ou `bounds` de la fenêtre sur les dimensions désirées. Lorsque vous définissez les limites, la taille et la position de la fenêtre peuvent changer simultanément. Toutefois, l'ordre de ces changements n'est pas garanti. Certains gestionnaires de fenêtres Linux ne permettent pas aux fenêtres de dépasser les limites de l'écran. Dans ce cas, la taille finale de la fenêtre peut être limitée du fait de l'ordre dans lequel les propriétés sont définies, même si l'impact des modifications aurait autrement résulté en une fenêtre valide. Par exemple, si vous modifiez à la fois la hauteur et la position y d'une fenêtre à proximité du bas de l'écran, la modification en pleine hauteur peut ne pas survenir lorsque la hauteur est modifiée avant la position y.

Remarque : *sous Linux, les propriétés des fenêtres sont modifiées de façon asynchrone. Si vous redimensionnez une fenêtre sur une ligne de votre programme et que vous lisez les dimensions à la ligne suivante, la modification n'est pas répercutée. Sur toutes les plates-formes, l'objet NativeWindow distribue l'événement* `resize` *lors du redimensionnement de la fenêtre. Si vous devez exécuter une action basée sur les nouvelles dimensions ou le nouvel état de la fenêtre (mettre en forme les contrôles de la fenêtre, par exemple), faites systématiquement appel à un gestionnaire d'événement* `resize`*. Pour plus d'informations, voir « Ecoute des événements d'une fenêtre » à la page 951.*

Le mode d'échelle de la scène indique le comportement de la scène de la fenêtre et de son contenu lorsqu'une fenêtre est redimensionnée. N'oubliez pas que les modes d'échelle de la scène sont conçus pour les situations où l'application ne contrôle pas la taille ou le format de son espace d'affichage (notamment dans le cas d'un navigateur Web). En règle générale, vous obtenez les meilleurs résultats en définissant la propriété `scaleMode` de la scène sur `StageScaleMode.NO_SCALE`. Pour mettre à l'échelle le contenu d'une fenêtre, vous pouvez toujours définir les paramètres `scaleX` et `scaleY` du contenu lors des modifications des limites de la fenêtre.

Déplacement d'une fenêtre

Pour déplacer une fenêtre sans la redimensionner, utilisez la méthode `startMove()` de la classe NativeWindow. A l'instar de la méthode `startResize()`, lorsque la méthode `startMove()` est appelée à partir d'un événement `mouseDown`, le déplacement est effectué avec la souris et se termine lorsque le système d'exploitation reçoit un événement `mouseUp`.

Pour plus d'informations sur les méthodes `startResize()` et `startMove()`, voir le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Pour déplacer une fenêtre par programmation, définissez les propriétés `x`, `y` ou `bounds` de la fenêtre sur la position requise. Lorsque vous définissez les limites, la taille et la position de la fenêtre peuvent changer simultanément.

Remarque : *sous Linux, les propriétés des fenêtres sont modifiées de façon asynchrone. Si vous déplacez une fenêtre sur une ligne de votre programme et que vous lisez la position à la ligne suivante, la modification n'est pas répercutée. Sur toutes les plates-formes, l'objet NativeWindow distribue l'événement* `move` *lors du repositionnement de la fenêtre. Si vous devez exécuter une action basée sur la nouvelle position de la fenêtre, faites systématiquement appel à un gestionnaire d'événement* `move`*. Pour plus d'informations, voir « Ecoute des événements d'une fenêtre » à la page 951.*

Exemple : Redimensionnement et déplacement des fenêtres

Adobe AIR 1.0 et les versions ultérieures

L'exemple suivant montre comment lancer les opérations de redimensionnement et de déplacement sur une fenêtre :

```
package
{
    import flash.display.Sprite;
    import flash.events.MouseEvent;
    import flash.display.NativeWindowResize;

    public class NativeWindowResizeExample extends Sprite
    {
        public function NativeWindowResizeExample():void
        {
            // Fills a background area.
            this.graphics.beginFill(0xFFFFFF);
            this.graphics.drawRect(0, 0, 400, 300);
            this.graphics.endFill();

            // Creates a square area where a mouse down will start the resize.
            var resizeHandle:Sprite =
                createSprite(0xCCCCCC, 20, this.width - 20, this.height - 20);
            resizeHandle.addEventListener(MouseEvent.MOUSE_DOWN, onStartResize);

            // Creates a square area where a mouse down will start the move.
            var moveHandle:Sprite = createSprite(0xCCCCCC, 20, this.width - 20, 0);
            moveHandle.addEventListener(MouseEvent.MOUSE_DOWN, onStartMove);
        }

        public function createSprite(color:int, size:int, x:int, y:int):Sprite
        {
            var s:Sprite = new Sprite();
            s.graphics.beginFill(color);
            s.graphics.drawRect(0, 0, size, size);
            s.graphics.endFill();
            s.x = x;
            s.y = y;
            this.addChild(s);
            return s;
        }

        public function onStartResize(event:MouseEvent):void
        {
            this.stage.nativeWindow.startResize(NativeWindowResize.BOTTOM_RIGHT);
        }

        public function onStartMove(event:MouseEvent):void
        {
            this.stage.nativeWindow.startMove();
        }
    }
}
```

Ecoute des événements d'une fenêtre

Adobe AIR 1.0 et les versions ultérieures

Pour écouter les événements distribués par une fenêtre, enregistrez un écouteur avec l'occurrence de la fenêtre. Par exemple, pour écouter un événement closing, enregistrez un écouteur avec la fenêtre, comme suit :

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

Lorsqu'un événement est distribué, la propriété `target` référence la fenêtre qui envoie l'événement.

La plupart des événements de fenêtre sont associés à deux messages. Le premier message signale l'imminence d'une modification au niveau de la fenêtre (modification pouvant être annulée) ; le second message indique que la modification a été effectuée. Par exemple, lorsqu'un utilisateur clique sur le bouton de fermeture d'une fenêtre, le message de l'événement closing est distribué. Si aucun écouteur n'annule l'événement, la fenêtre se ferme et l'événement close est distribué à l'un des écouteurs.

En règle générale, les événements d'avertissement, tels que `closing`, ne sont distribués que lorsque le chrome système a été utilisé pour déclencher un événement. L'appel de la méthode `close()`, par exemple, ne distribue pas automatiquement l'événement `closing` ; seul l'événement `close` est distribué. Vous pouvez toutefois créer un événement closing et le distribuer à l'aide de la méthode `dispatchEvent()` de la fenêtre.

Les événements de fenêtre qui distribuent un objet Event sont les suivants :

Evénement	Description
activate	Distribué lorsque la fenêtre reçoit le focus.
deactivate	Distribué lorsque la fenêtre perd le focus.
closing	Distribué lorsque la fenêtre est sur le point de se fermer. Cet événement ne se produit automatiquement que lorsque l'utilisateur appuie sur le bouton de fermeture du chrome système ou, sous OS X, lorsque la commande Quitter est invoquée.
close	Distribué lorsque la fenêtre a été fermée.

Les événements de fenêtre qui distribuent un objet NativeWindowBoundsEvent sont les suivants :

Evénement	Description
moving	Distribué immédiatement avant que le coin supérieur gauche de la fenêtre change de position, suite au déplacement, au redimensionnement ou à la modification de l'état d'affichage de la fenêtre.
move	Distribué après que le coin supérieur gauche de la fenêtre a changé de position.
resizing	Distribué immédiatement avant que la largeur ou la hauteur de la fenêtre change, suite au redimensionnement ou à la modification de l'état d'affichage de la fenêtre.
resize	Distribué après que la taille de la fenêtre a changé.

Pour les événements NativeWindowBoundsEvent, vous pouvez utiliser les propriétés `beforeBounds` et `afterBounds` pour déterminer les limites de la fenêtre avant une modification imminente ou après avoir effectué une modification.

Les événements de fenêtre qui distribuent un objet NativeWindowDisplayStateEvent sont les suivants :

Evénement	Description
displayStateChanging	Distribué immédiatement avant que l'état d'affichage de la fenêtre change.
displayStateChange	Distribué une fois l'état d'affichage de la fenêtre modifié.

Pour les événements NativeWindowDisplayStateEvent, vous pouvez utiliser les propriétés `beforeDisplayState` et `afterDisplayState` pour déterminer l'état d'affichage de la fenêtre avant une modification imminente et après avoir effectué une modification.

Dans certains gestionnaires de fenêtres Linux, aucun événement de modification de l'état d'affichage n'est distribué lors de l'agrandissement d'une fenêtre possédant un paramètre de taille maximale. (La fenêtre est définie sur l'état d'affichage maximal, mais n'est pas redimensionnée.)

Affichage des fenêtres en mode plein écran

Adobe AIR 1.0 et les versions ultérieures

La définition de la propriété `displayState` de la scène sur `StageDisplayState.FULL_SCREEN_INTERACTIVE` met la fenêtre en mode plein écran et la saisie clavier *is* est autorisée dans ce mode. (Pour le contenu SWF s'exécutant dans un navigateur, la saisie clavier n'est pas autorisée). Pour quitter le mode plein écran, l'utilisateur doit appuyer sur la touche Echap.

Remarque : *certains gestionnaires de fenêtres Linux ne modifient pas les dimensions de la fenêtre pour remplir l'écran si une taille maximale est définie pour celle-ci (mais suppriment le chrome système de la fenêtre).*

Par exemple, le code Flex suivant définit une application AIR simple qui configure un terminal d'écran complet :

```
 <?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    applicationComplete="init()" backgroundColor="0x003030" focusRect="false">
    <mx:Script>
        <![CDATA[
            private function init():void
            {
                stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
                focusManager.setFocus(terminal);
                terminal.text = "Welcome to the dumb terminal app. Press the ESC key to
exit..\n";
                terminal.selectionBeginIndex = terminal.text.length;
                terminal.selectionEndIndex = terminal.text.length;
            }
        ]]>
    </mx:Script>
    <mx:TextArea
        id="terminal"
        height="100%" width="100%"
        scroll="false"
        backgroundColor="0x003030"
        color="0xCCFF00"
        fontFamily="Lucida Console"
        fontSize="44"/>
</mx:WindowedApplication>
```

L'exemple ActionScript suivant pour Flash simule un terminal d'écran complet simple :

```
    import flash.display.Sprite;
    import flash.display.StageDisplayState;
    import flash.text.TextField;
    import flash.text.TextFormat;

    public class FullScreenTerminalExample extends Sprite
    {
        public function FullScreenTerminalExample():void
        {
            var terminal:TextField = new TextField();
            terminal.multiline = true;
            terminal.wordWrap = true;
            terminal.selectable = true;
            terminal.background = true;
            terminal.backgroundColor = 0x00333333;

            this.stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;

            addChild(terminal);
            terminal.width = 550;
            terminal.height = 400;

            terminal.text = "Welcome to the dumb terminal application. Press the ESC key to
exit.\n_";

            var tf:TextFormat = new TextFormat();
            tf.font = "Courier New";
            tf.color = 0x00CCFF00;
            tf.size = 12;
            terminal.setTextFormat(tf);

            terminal.setSelection(terminal.text.length - 1, terminal.text.length);
        }
    }
```

Chapitre 53 : Ecrans dans AIR

Adobe AIR 1.0 et les versions ultérieures

La classe Screen d'Adobe® AIR® permet d'accéder à des informations sur les écrans rattachés à un ordinateur ou à un périphérique.

Voir aussi

flash.display.Screen

Principes de base des écrans dans AIR

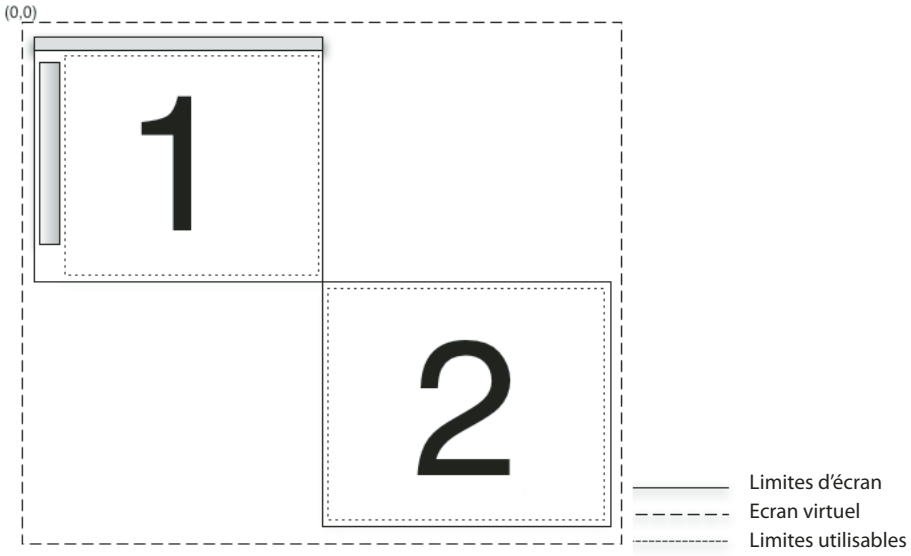
Adobe AIR 1.0 et les versions ultérieures

- Mesure de l'ordinateur de bureau virtuel (Flex)
- Mesure de l'ordinateur de bureau virtuel (Flash)

L'interface de programmation de l'écran contient une seule classe, Screen, qui permet à des membres statiques de lire des informations système sur les écrans et à des membres d'occurrences de décrire un écran particulier.

Plusieurs écrans peuvent être reliés à un système d'ordinateurs, ce qui peut donner lieu à un espace virtuel qui contient plusieurs écrans d'ordinateur. La classe Screen d'AIR fournit des informations sur les écrans, leur organisation relative et leur espace utilisable. Si plusieurs moniteurs mappent sur le même écran, seul un écran existe. Si la taille d'un écran est plus grande que la zone d'affichage du moniteur, il n'y a pas moyen de savoir quelle partie de l'écran est actuellement visible.

Un écran représente une zone d'affichage indépendante de l'ordinateur. Les écrans sont décrits comme des rectangles au sein d'un ordinateur de bureau virtuel. Le coin supérieur gauche de l'écran désigné comme zone d'affichage principale constitue l'origine du système de coordonnées de l'ordinateur virtuel. Toutes les valeurs utilisées dans la description d'un écran sont exprimées en pixels.



Dans cette organisation d'écrans, deux écrans existent sur l'ordinateur virtuel. Les coordonnées du coin supérieur gauche de l'écran principal (nº 1) sont toujours (0,0). Si l'organisation des écrans change pour désigner l'écran nº 2 comme écran principal, les coordonnées de l'écran nº 1 deviennent négatives. Les barres de menus, les barres de tâches et les Docks sont ignorés lorsque les limites utilisables sont signalées pour un écran.

Pour plus d'informations sur la classe, les méthodes, les propriétés et les événements de l'interface de programmation d'écran, voir Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Dénombrement des écrans

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez dénombrer les écrans d'un ordinateur de bureau virtuel avec les méthodes et propriétés suivantes :

Méthode ou propriété	Description
Screen.screens	Fournit un tableau d'objets Screen qui décrit les écrans disponibles. L'ordre du tableau n'a pas d'importance.
Screen.mainScreen	Fournit un objet Screen pour l'écran principal. Sous Mac OS X, l'écran principal est celui qui affiche la barre de menus. Sous Windows, l'écran principal est celui désigné par le système.
Screen.getScreensForRectangle()	Fournit un tableau d'objets Screen qui décrit les écrans qui sont coupés par le rectangle donné. Le rectangle transmis par cette méthode est défini par des coordonnées, exprimées en pixels, sur l'ordinateur virtuel. S'il n'y a aucune intersection entre écrans et rectangle, le tableau est vide. Vous pouvez utiliser cette méthode pour trouver sur quels écrans une fenêtre est affichée.

N'enregistrez pas les valeurs renvoyées par les méthodes et propriétés de la classe Screen. L'utilisateur ou le système d'exploitation peuvent changer les écrans disponibles et leur organisation en tout temps.

L'exemple suivant utilise l'interface de programmation des écrans pour déplacer une fenêtre entre plusieurs écrans en réponse à la pression sur les touches de direction. Pour déplacer la fenêtre sur l'écran suivant, l'exemple lit le tableau `screens` et le trie verticalement ou horizontalement, suivant la touche de direction sur laquelle on a appuyé. Le code parcourt le tableau trié en comparant chaque écran aux coordonnées de l'écran actif. Pour identifier l'écran actif de la fenêtre, l'exemple appelle `Screen.getScreensForRectangle()` en le transmettant dans les limites de la fenêtre.

```
package {
    import flash.display.Sprite;
    import flash.display.Screen;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;

    public class ScreenExample extends Sprite
    {
        public function ScreenExample()
        {
                stage.align = StageAlign.TOP_LEFT;
                stage.scaleMode = StageScaleMode.NO_SCALE;

                stage.addEventListener(KeyboardEvent.KEY_DOWN,onKey);
        }

        private function onKey(event:KeyboardEvent):void{
            if(Screen.screens.length > 1){
                switch(event.keyCode){
                    case Keyboard.LEFT :
                        moveLeft();
                        break;
                    case Keyboard.RIGHT :
                        moveRight();
                        break;
                    case Keyboard.UP :
                        moveUp();
                        break;
                    case Keyboard.DOWN :
                        moveDown();
                        break;
                }
            }
        }

        private function moveLeft():void{
            var currentScreen = getCurrentScreen();
            var left:Array = Screen.screens;
            left.sort(sortHorizontal);
            for(var i:int = 0; i < left.length - 1; i++){
                if(left[i].bounds.left < stage.nativeWindow.bounds.left){
                    stage.nativeWindow.x +=
                        left[i].bounds.left - currentScreen.bounds.left;
                    stage.nativeWindow.y += left[i].bounds.top - currentScreen.bounds.top;
                }
            }
        }

        private function moveRight():void{
            var currentScreen:Screen = getCurrentScreen();
            var left:Array = Screen.screens;
            left.sort(sortHorizontal);
            for(var i:int = left.length - 1; i > 0; i--){
                if(left[i].bounds.left > stage.nativeWindow.bounds.left){
                    stage.nativeWindow.x +=
```

```
                    left[i].bounds.left - currentScreen.bounds.left;
                stage.nativeWindow.y += left[i].bounds.top - currentScreen.bounds.top;
            }
        }
    }

    private function moveUp():void{
        var currentScreen:Screen = getCurrentScreen();
        var top:Array = Screen.screens;
        top.sort(sortVertical);
        for(var i:int = 0; i < top.length - 1; i++){
            if(top[i].bounds.top < stage.nativeWindow.bounds.top){
                stage.nativeWindow.x += top[i].bounds.left - currentScreen.bounds.left;
                stage.nativeWindow.y += top[i].bounds.top - currentScreen.bounds.top;
                break;
            }
        }
    }

    private function moveDown():void{
        var currentScreen:Screen = getCurrentScreen();

        var top:Array = Screen.screens;
        top.sort(sortVertical);
        for(var i:int = top.length - 1; i > 0; i--){
            if(top[i].bounds.top > stage.nativeWindow.bounds.top){
                stage.nativeWindow.x += top[i].bounds.left - currentScreen.bounds.left;
                stage.nativeWindow.y += top[i].bounds.top - currentScreen.bounds.top;
                break;
            }
        }
    }

    private function sortHorizontal(a:Screen,b:Screen):int{
        if (a.bounds.left > b.bounds.left){
            return 1;
        } else if (a.bounds.left < b.bounds.left){
            return -1;
        } else {return 0;}
    }

    private function sortVertical(a:Screen,b:Screen):int{
        if (a.bounds.top > b.bounds.top){
            return 1;
        } else if (a.bounds.top < b.bounds.top){
            return -1;
        } else {return 0;}
    }

    private function getCurrentScreen():Screen{
        var current:Screen;
        var screens:Array = Screen.getScreensForRectangle(stage.nativeWindow.bounds);
        (screens.length > 0) ? current = screens[0] : current = Screen.mainScreen;
        return current;
    }
  }
}
```

Chapitre 54 : Impression

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les moteurs d'exécution Flash (tels qu'Adobe® Flash® Player et Adobe® AIR™) peuvent communiquer avec l'interface d'impression du système d'exploitation, ce qui vous permet de transmettre des pages au spooler de l'imprimante. Chaque page envoyée au spooler peut comprendre du contenu visible, dynamique ou invisible pour l'utilisateur à l'écran, y compris des valeurs de base de données et du texte dynamique. Par ailleurs, les propriétés de la classe flash.printing.PrintJob contiennent des valeurs basées sur les paramètres d'impression de l'utilisateur, afin d'assurer le formatage correct des pages.

Les stratégies d'utilisation des méthodes et propriétés de la classe flash.printing.PrintJob sont passées en revue ci-après. Elles permettent de créer une tâche d'impression, de lire les paramètres d'impression d'un utilisateur et d'ajuster une tâche d'impression en fonction des informations renvoyées par un moteur d'exécution de Flash et le système d'exploitation de l'utilisateur.

Principes de base de l'impression

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans ActionScript 3.0, vous utilisez la classe PrintJob pour créer des instantanés de contenu d'affichage à convertir en représentation encre-et-papier dans une impression. Dans certains cas, définir le contenu à imprimer revient à le définir pour un affichage à l'écran—vous positionnez et dimensionnez des éléments pour créer la mise en forme souhaitée. Néanmoins, l'impression a des caractéristiques qui la différencient de la mise en forme à l'écran. Par exemple, les imprimantes utilisent une résolution différente des écrans d'ordinateur. Le contenu d'un écran d'ordinateur est dynamique et peut changer, alors que le contenu imprimé est statique. Lorsque vous planifiez une impression, tenez compte des contraintes de taille de page fixe et de la possibilité d'imprimer plusieurs pages.

Aussi évidentes que puissent paraître ces différences, il est important de les garder à l'esprit lors de la configuration de l'impression avec ActionScript. Une impression précise repose sur la combinaison des valeurs stipulées par vous et des caractéristiques de l'imprimante de l'utilisateur. Les propriétés de la classe PrintJob permettent de déterminer les caractéristiques importantes de l'imprimante de l'utilisateur.

Concepts importants et terminologie

La liste de référence suivante contient des termes importants relatifs à l'impression :

Spouleur Partie du système d'exploitation ou du logiciel du pilote d'imprimante qui effectue le suivi des pages en attente d'impression et les envoie à l'imprimante dès qu'elle est disponible.

Orientation de page Rotation du contenu imprimé par rapport au papier (horizontale - paysage ou verticale - portrait).

Tâche d'impression Page ou jeu de pages constituant une seule impression.

Impression d'une page

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous utilisez une occurrence de la classe PrintJob pour gérer l'impression. Pour imprimer une page de base dans Flash Player ou AIR, vous utilisez quatre instructions à la suite :

- `new PrintJob()` : crée une occurrence de tâche d'impression dotée du nom que vous spécifiez.
- `PrintJob.start()` : initialise le processus d'impression pour le système d'exploitation (en appelant la boîte de dialogue d'impression destinée à l'utilisateur) et définit les propriétés en lecture seule de la tâche d'impression.
- `PrintJob.addPage()` : comprend les détails du contenu de la tâche d'impression, notamment l'objet Sprite (et ses éventuels enfants), la taille de la zone d'impression et le mode d'impression d'image (vectoriel ou bitmap) utilisé par l'imprimante. Vous pouvez effectuer plusieurs appels successifs de `addPage()` afin d'imprimer plusieurs sprite sur plusieurs pages.
- `PrintJob.send()` : envoie les pages à l'imprimante du système d'exploitation.

Exemple de script de tâche d'impression simple, qui comporte des instructions `package`, `import` et `class` à des fins de compilation :

```
package
{
    import flash.printing.PrintJob;
    import flash.display.Sprite;

    public class BasicPrintExample extends Sprite
    {
        var myPrintJob:PrintJob = new PrintJob();
        var mySprite:Sprite = new Sprite();

        public function BasicPrintExample()
        {
            myPrintJob.start();
            myPrintJob.addPage(mySprite);
            myPrintJob.send();
        }
    }
}
```

Remarque : *cet exemple vise à illustrer les éléments de base d'un script de tâche d'impression et ne contient aucun dispositif de gestion des erreurs. Pour construire un script qui réponde convenablement à l'annulation d'une tâche d'impression par l'utilisateur, voir «* *Utilisation des exceptions et des renvois* *» à la page 960.*

S'il s'avère nécessaire de purger les propriétés d'un objet PrintJob, définissez la variable PrintJob sur `null` (par exemple `myPrintJob = null`).

Tâches des moteurs d'exécution de Flash et impression système

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les moteurs d'exécution de Flash transmettent des pages à l'interface d'impression du système d'exploitation ; veillez donc à identifier les tâches gérées par les moteurs d'exécution de Flash et celles gérées par l'interface d'impression du système d'exploitation. Les moteurs d'exécution de Flash peuvent initialiser une tâche d'impression, lire certains des paramètres de page de l'imprimante, transmettre le contenu d'une tâche d'impression au système d'exploitation et vérifier si l'utilisateur ou le système annule une tâche. D'autres processus, tels que l'affichage de boîtes de dialogue spécifiques à l'imprimante, l'annulation d'une tâche d'impression mise en file d'attente ou l'indication de l'état de l'imprimante, sont supervisés par le système d'exploitation. Si les moteurs d'exécution de Flash sont capables de réagir en cas de problème d'initialisation ou de formatage d'une tâche d'impression, ils peuvent uniquement signaler certaines propriétés ou conditions transmises par l'interface d'impression du système d'exploitation. C'est le rôle du développeur d'élaborer un code susceptible de répondre à ces propriétés ou conditions.

Utilisation des exceptions et des renvois

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Avant d'appeler `addPage()` et `send()`, vérifiez que la méthode `PrintJob.start()` renvoie la valeur `true`, au cas où l'utilisateur aurait annulé la tâche d'impression. Une manière simple de vérifier si ces méthodes ont été annulées avant de poursuivre consiste à les intégrer à une instruction `if`, comme suit :

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

Si `PrintJob.start()` est défini sur `true`, l'utilisateur a sélectionné Imprimer ou un moteur d'exécution de Flash, tel que Flash Player ou AIR, a initié une commande d'impression. Vous pouvez alors appeler les méthodes `addPage()` et `send()`.

En outre, pour faciliter la gestion du processus d'impression, les moteurs d'exécution Flash renvoient des exceptions pour la méthode `PrintJob.addPage()`. Vous pouvez ainsi intercepter les erreurs et fournir des informations et des options à l'utilisateur. Si une méthode `PrintJob.addPage()` échoue, il est également possible d'appeler une autre fonction ou d'arrêter la tâche d'impression en cours. Vous interceptez ces exceptions en intégrant des appels `addPage()` à une instruction `try..catch`, comme illustré par l'exemple suivant. Dans cet exemple, `[params]` constitue un espace réservé pour les paramètres spécifiant le contenu réel à imprimer :

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

Une fois la tâche d'impression lancée, vous pouvez ajouter le contenu à l'aide de `PrintJob.addPage()` pour voir s'il génère une exception (par exemple si l'utilisateur a annulé la tâche d'impression). Si tel est le cas, vous pouvez soit ajouter une logique à l'instruction `catch` pour fournir à l'utilisateur (ou au moteur d'exécution Flash) des informations et des options, soit arrêter la tâche d'impression en cours. Si l'ajout de page réussit, vous pouvez procéder à l'envoi des pages vers l'imprimante à l'aide de `PrintJob.send()`.

Si le moteur d'exécution Flash rencontre un problème lors de l'envoi de la tâche d'impression à l'imprimante (par exemple, si l'imprimante est hors ligne), il est également possible d'intercepter cette exception. Vous pouvez alors proposer d'autres informations ou options (tel l'affichage d'un message ou le lancement d'une alerte dans une animation). Vous pouvez, par exemple, affecter un nouveau texte à un champ texte dans une instruction `if..else`, comme illustré par le code suivant :

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

Pour disposer d'un exemple qui fonctionne, voir « Exemple d'impression : mise à l'échelle, recadrage et ajustement » à la page 969.

Utilisation des propriétés de page

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur clique sur OK dans la boîte de dialogue d'impression et que `PrintJob.start()` renvoie `true`, vous pouvez accéder aux propriétés définies par les paramètres de l'imprimante. Il s'agit notamment de la largeur et de la hauteur du papier (`pageHeight` et `pageWidth`) et de l'orientation du contenu sur la feuille. Puisque ces paramètres d'impression ne sont pas contrôlés par le moteur d'exécution Flash, il est impossible de les modifier. Vous pouvez en revanche les utiliser pour faire correspondre le contenu à envoyer à l'imprimante aux paramètres en cours. Pour plus d'informations, voir « Définition de la taille, de l'échelle et de l'orientation » à la page 963.

Définition du rendu vectoriel ou bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous avez la possibilité de définir manuellement la tâche d'impression de manière à transmettre chaque page sous forme d'image vectorielle ou bitmap. Dans certains cas, l'impression vectorielle produit un fichier de file d'attente plus réduit et une image de meilleure qualité que l'impression bitmap. Toutefois, si le contenu inclut une image bitmap et que vous souhaitez préserver la transparence alpha ou tout autre effet de couleur, imprimez la page en tant qu'image bitmap. En outre, une imprimante non-PostScript convertit automatiquement les graphiques vectoriels en images bitmap.

Vous spécifiez l'impression bitmap en transmettant un objet PrintJobOptions en tant que troisième paramètre de `PrintJob.addPage()`.

Dans Flash Player et les versions d'AIR antérieures à 2, définissez le paramètre `printAsBitmap` de l'objet PrintJobOptions sur `true`, comme suit :

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

Si vous ne spécifiez aucune valeur pour le troisième paramètre, la tâche d'impression utilise le paramètre par défaut, soit l'impression vectorielle.

Pour AIR 2 et les versions ultérieures, faites appel à la propriété `printMethod` de l'objet PrintJobOptions pour spécifier la méthode d'impression. Cette propriété gère trois valeurs, qui sont définies en tant que constantes dans la classe PrintMethod :

- `PrintMethod.AUTO` : choisit automatiquement la méthode d'impression la plus adaptée en fonction du contenu imprimé. Ainsi, si une page ne contient que du texte, la méthode d'impression vectorielle est sélectionnée. Toutefois, si une image en filigrane avec transparence Alpha est superposée au texte, l'impression bitmap est choisie pour conserver la transparence.
- `PrintMethod.BITMAP` : impose l'impression bitmap quel que soit le contenu.
- `PrintMethod.VECTOR` : impose l'impression vectorielle quel que soit le contenu.

Temporisation des instructions de tâche d'impression

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Contrairement aux versions précédentes, ActionScript 3.0 ne limite pas un objet PrintJob à une image unique. Cependant, comme le système d'exploitation indique l'état de l'impression après que l'utilisateur a cliqué sur le bouton OK dans la boîte de dialogue d'impression, appelez `PrintJob.addPage()` et `PrintJob.send()` dès que possible pour envoyer les pages au spouleur. Si l'accès à l'image contenant l'appel `PrintJob.send()` est soumis à un délai, le processus d'impression est également retardé.

Dans ActionScript 3.0, le délai de script est de 15 secondes. Par conséquent, l'intervalle entre chaque instruction essentielle de la séquence de tâche d'impression ne peut pas dépasser 15 secondes. En d'autres termes, la limite de 15 secondes concerne les intervalles suivants :

- Entre `PrintJob.start()` et la première instruction `PrintJob.addPage()`
- Entre `PrintJob.addPage()` et l'instruction suivante `PrintJob.addPage()`
- Entre la dernière instruction `PrintJob.addPage()` et `PrintJob.send()`

Si l'un des intervalles ci-dessus excède 15 secondes, l'appel suivant de la méthode `PrintJob.start()` pour l'occurrence de PrintJob renvoie `false` et l'appel suivant de la méthode `PrintJob.addPage()` pour l'occurrence de PrintJob entraîne le renvoi d'une exception d'exécution par Flash Player ou AIR.

Définition de la taille, de l'échelle et de l'orientation

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La section « Impression d'une page » à la page 959 décrit en détail les étapes d'une tâche d'impression de base, dans laquelle la sortie reproduit directement le sprite spécifié, selon sa taille d'affichage et sa position à l'écran. Néanmoins, les imprimantes utilisent différentes résolutions d'impression et certains de leurs paramètres peuvent altérer l'aspect du sprite imprimé.

Les moteurs d'exécution Flash peuvent lire les paramètres d'impression du système d'exploitation, mais notez que ces propriétés sont accessibles en lecture seule. En d'autres termes, bien que vous puissiez réagir à leur valeur, il est possible de les définir. Par exemple, il est possible de déterminer le paramètre de format de page de l'imprimante et d'ajuster votre contenu en fonction. Vous pouvez de même identifier les paramètres de marge de l'imprimante ainsi que l'orientation des pages. Pour répondre aux paramètres de l'imprimante, spécifiez une zone d'impression, effectuez un ajustement pour tenir compte de la différence entre la résolution de l'écran et la mesure de points de l'imprimante, ou faites correspondre le contenu aux paramètres de taille et d'orientation de l'imprimante de l'utilisateur.

Définition de la zone d'impression à l'aide de rectangle

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `PrintJob.addPage()` vous permet de spécifier la partie du sprite que vous souhaitez imprimer. Le deuxième paramètre, `printArea` prend la forme d'un objet Rectangle. Vous pouvez fournir la valeur de ce paramètre de trois manières :

- Créez un objet Rectangle doté de propriétés spécifiques, puis utilisez-le dans l'appel `addPage()`, comme dans l'exemple suivant :

```
private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
myPrintJob.addPage(sheet, rect1);
```

- Si vous n'avez pas encore spécifié l'objet Rectangle, vous pouvez le faire dans l'appel lui-même, comme illustré ci-dessous :

```
myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
```

- Si vous prévoyez de fournir des valeurs pour le troisième paramètre de l'appel `addPage()` sans pour autant spécifier un objet Rectangle, utilisez la valeur `null` pour le deuxième paramètre, comme suit :

```
myPrintJob.addPage(sheet, null, options);
```

Comparaison entre les points et les pixels

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La largeur et la hauteur d'un rectangle sont exprimées en pixels. Une imprimante utilise les points en tant qu'unités de mesure. Les points ont une taille physique fixe (1/72e de pouce), mais la taille d'un pixel à l'écran varie selon la résolution de ce dernier. De ce fait, le taux de conversion entre les pixels et les points dépend de la configuration de l'imprimante et du redimensionnement éventuel du sprite. Un sprite non redimensionné d'une largeur de 72 pixels mesure un pouce (2,54 cm) de large lorsqu'il est imprimé, sachant qu'un point correspond à un pixel quelle que soit la résolution de l'écran.

Vous pouvez utiliser les équivalences suivantes pour convertir les pouces ou les centimètres en twips ou points (un twip correspond à 1/20ème de point) :

- 1 point = 1/72ème de pouce = 20 twips
- 1 pouce = 72 points = 1 440 twips
- 1 centimètre = 567 twips

Si vous omettez le paramètre `printArea` ou s'il est transmis de façon incorrecte, la zone entière du sprite est imprimée.

Redimensionnement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous souhaitez redimensionner un objet Sprite avant de l'imprimer, définissez les propriétés de redimensionnement (voir « Manipulation de la taille et de l'échelle des objets » à la page 186) avant d'appeler la méthode `PrintJob.addPage()`, puis rétablissez leurs valeurs d'origine après l'impression. L'échelle d'un objet Sprite ne dépend pas de la propriété `printArea`. En d'autres termes, si vous spécifiez une zone d'impression de 50 pixels par 50 pixels, 2 500 pixels sont imprimés. Si vous redimensionnez l'objet Sprite, les 2 500 pixels sont imprimés, mais l'objet est imprimé à l'échelle retenue.

A titre d'exemple, voir « Exemple d'impression : mise à l'échelle, recadrage et ajustement » à la page 969.

Impression en mode paysage ou portrait

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player et AIR étant capables de détecter les paramètres d'orientation, vous pouvez insérer dans votre code ActionScript une logique permettant d'ajuster la taille du contenu ou son orientation en fonction des paramètres de l'imprimante, comme illustré dans l'exemple ci-après.

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

Remarque : *si vous envisagez de lire le paramètre système pour définir l'orientation du contenu sur le papier, n'oubliez pas d'importer la classe PrintJobOrientation. La classe PrintJobOrientation fournit des valeurs constantes qui définissent l'orientation du contenu sur la page. Vous importez la classe à l'aide de l'instruction suivante :*

```
import flash.printing.PrintJobOrientation;
```

Ajustement de la hauteur et de la largeur au format du papier

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par l'utilisation d'une stratégie semblable à la gestion des paramètres d'orientation de l'imprimante, vous pouvez lire les paramètres de hauteur et de largeur de page, puis en tenir compte en intégrant une logique dans une instruction `if`. Le code suivant illustre ce cas de figure :

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

En outre, il est possible de déterminer les paramètres de marge d'une page en comparant les dimensions de cette page à celle du papier, comme illustré ci-après :

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

Techniques d'impression avancées

Adobe AIR 2 et ultérieur

Depuis la version 2 d'Adobe AIR, la classe PrintJob comporte d'autres propriétés et méthodes et trois classes supplémentaires, PrintUIOptions, PaperSize et PrintMethod, sont prises en charge. Ces modifications gèrent d'autres flux de travail d'impression et permettent aux créateurs de mieux contrôler le processus d'impression. Parmi les modifications figurent :

- Boîtes de dialogue de mise en page : affichage de boîtes de dialogue de mise en page standard et personnalisée. L'utilisateur peut définir les étendues de pages, le format du papier, l'orientation et la mise à l'échelle avant de lancer l'impression.
- Affichage avant l'impression : permet de créer un mode d'affichage qui indique avec précision le format du papier, les marges et la position du contenu sur la page.
- Impression restreinte : les créateurs peuvent restreindre les options d'impressions, telles que l'étendue de pages imprimables.
- Options relatives à la qualité : les créateurs peuvent ajuster la qualité d'impression d'un document et autoriser l'utilisateur à sélectionner les options de résolution et de couleur.
- Sessions d'impression multiples : il est à présent possible d'utiliser une occurrence unique de PrintJob pour gérer plusieurs sessions d'impression. Les applications peuvent proposer des paramètres cohérents chaque fois que les boîtes de dialogue de mise en page et d'impression s'affichent.

Modifications du flux de travaux d'impression

Le nouveau flux de travaux d'impression se compose des étapes suivantes :

- `new PrintJob()` : crée une occurrence de PrintJob (ou réutilise une occurrence existante). Un grand nombre de méthodes et de propriétés PrintJob, telles que `selectPaperSize()`, sont disponibles avant le lancement de la tâche d'impression ou lors de l'impression.
- `PrintJob.showPageSetupDialog()` : (facultatif) affiche la boîte de dialogue de mise en page sans démarrer de tâche d'impression.
- `PrintJob.start()` ou `PrintJob.start2()` : parallèlement à la méthode `start()`, la méthode `start2()` permet de lancer le processus de mise en file d'attente d'impression. La méthode `start2()` permet d'activer l'affichage de la boîte de dialogue d'impression et, le cas échéant, de la personnaliser.
- `PrintJob.addPage()` : ajoute un contenu à la tâche d'impression. Cette étape est identique à celle du processus existant.
- `PrintJob.send()` ou `PrintJob.terminate()` : envoie les pages à l'imprimante sélectionnée ou met fin à la tâche d'impression sans l'envoyer. Les tâches d'impression sont arrêtées suite à une erreur. Si une occurrence de PrintJob est arrêtée, elle peut être réutilisée. Que la tâche d'impression soit envoyée à l'imprimante ou arrêtée, les paramètres d'impression actifs sont conservés lorsque vous réutilisez l'occurrence de PrintJob.

Boîte de dialogue de mise en page

La méthode `showPageSetupDialog()` affiche la boîte de dialogue de mise en page du système d'exploitation si l'environnement actif le permet. Vérifiez toujours la propriété `supportsPageSetupDialog` avant d'appeler cette méthode. Voici un exemple simple :

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

Si besoin est, vous pouvez associer la méthode à une propriété de la classe PrintUIOptions pour contrôler les options affichées dans la boîte de dialogue de mise en page. Vous pouvez définir le nombre minimal et maximal de pages. L'exemple suivant restreint l'impression aux trois premières pages :

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
if (PrintJob.supportsPageSetupDialog) {
    var uiOpt:PrintUIOptions = new PrintUIOptions();
    uiOpt.minPage = 1;
    uiOpt.maxPage = 3;
    myPrintJob.showPageSetupDialog(uiOpt);
}
```

Modification des paramètres d'impression

Vous pouvez modifier les paramètres d'une occurrence de PrintJob à tout moment après sa création. Il est ainsi possible de modifier les paramètres entre des appels d'`addPage()` et après l'envoi ou l'arrêt d'une tâche d'impression. Certains paramètres, tels que la propriété `printer`, s'appliquent à la tâche d'impression entière et non à des pages spécifiques. Ils doivent être définis avant d'appeler `start()` ou `start2()`.

La méthode `selectPaperSize()` permet de définir le format du papier par défaut dans les boîtes de dialogue de mise en page et d'impression. Vous pouvez également l'appeler au cours d'une tâche d'impression pour définir le format du papier d'une étendue de pages. Vous utilisez à cet effet des constantes définies dans la classe `PaperSize`; comme illustré par l'exemple suivant, qui sélectionne une enveloppe de format 10 :

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

La propriété `printer` permet d'extraire ou de définir le nom de l'imprimante associée à la tâche d'impression active. Par défaut, elle est définie sur le nom de l'imprimante par défaut. La propriété `printer` est définie sur `null` si aucune imprimante n'est disponible ou si le système ne prend pas en charge l'impression. Pour changer d'imprimante, commencez par obtenir la liste des imprimantes disponibles par le biais de la propriété `printers`. Cette propriété est un objet Vector dont les éléments String correspondent aux noms d'imprimantes disponibles. Définissez la propriété `printer` sur l'une de ces valeurs String pour activer l'imprimante correspondante. Il est impossible de modifier la propriété `printer` d'une tâche d'impression active. Toute tentative de modification de cette propriété après qu'un appel de `start()` ou `start2()` a abouti ou avant l'envoi ou l'arrêt de la tâche d'impression échoue. Exemple de définition de la propriété :

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

La propriété `copies` extrait la valeur correspondant au nombre d'exemplaires défini dans la boîte de dialogue d'impression du système d'exploitation. Les propriétés `firstPage` et `lastPage` extraient l'étendue de pages. La propriété `orientation` extrait le paramètre d'orientation de page. Vous pouvez définir ces propriétés de sorte à remplacer les valeurs saisies dans la boîte de dialogue d'impression. L'exemple suivant définit ces propriétés :

```
import flash.printing.PrintJob;
import flash.printing.PrintJobOrientation;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.copies = 3;
myPrintJob.firstPage = 1;
myPrintJob.lastPage = 3;
myPrintJob.orientation = PrintJobOrientation.LANDSCAPE;
```

Les paramètres en lecture seule suivants associés à `PrintJob` fournissent des informations pertinentes sur la configuration de l'imprimante active :

- `paperArea` : limites rectangulaires du support d'impression, exprimées en points.
- `printableArea` : limites rectangulaires de la zone imprimable, exprimées en points.
- `maxPixelsPerInch` : résolution physique de l'imprimante active, exprimée en pixel par pouce.
- `isColor` : capacité de l'imprimante active à imprimer les couleurs (renvoie `true` si l'imprimante active gère les couleurs).

Voir « Exemple d'impression : options d'impression et de mise en page » à la page 971

Exemple d'impression : impression de plusieurs pages

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous devez imprimer plusieurs pages de contenu, vous pouvez associer chaque page à un sprite différent (dans le cas présent, `sheet1` et `sheet2`). Utilisez ensuite `PrintJob.addPage()` pour chaque sprite. Le code suivant illustre cette technique :

```
package
{
    import flash.display.MovieClip;
    import flash.printing.PrintJob;
    import flash.printing.PrintJobOrientation;
    import flash.display.Stage;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.geom.Rectangle;

    public class PrintMultiplePages extends MovieClip
    {
        private var sheet1:Sprite;
        private var sheet2:Sprite;

        public function PrintMultiplePages():void
        {
            init();
            printPages();
        }

        private function init():void
        {
            sheet1 = new Sprite();
            createSheet(sheet1, "Once upon a time...", {x:10, y:50, width:80, height:130});
            sheet2 = new Sprite();
            createSheet(sheet2, "There was a great story to tell, and it ended quickly.\n\nThe
end.", null);
        }

        private function createSheet(sheet:Sprite, str:String, imgValue:Object):void
        {
            sheet.graphics.beginFill(0xEEEEEE);
            sheet.graphics.lineStyle(1, 0x000000);
            sheet.graphics.drawRect(0, 0, 100, 200);
            sheet.graphics.endFill();

            var txt:TextField = new TextField();
            txt.height = 200;
            txt.width = 100;
            txt.wordWrap = true;
            txt.text = str;

            if (imgValue != null)
            {
                var img:Sprite = new Sprite();
                img.graphics.beginFill(0xFFFFFF);
                img.graphics.drawRect(imgValue.x, imgValue.y, imgValue.width, imgValue.height);
                img.graphics.endFill();
                sheet.addChild(img);
            }
            sheet.addChild(txt);
        }

        private function printPages():void
        {
            var pj:PrintJob = new PrintJob();
```

```
            var pagesToPrint:uint = 0;
            if (pj.start())
            {
                if (pj.orientation == PrintJobOrientation.LANDSCAPE)
                {
                    throw new Error("Page is not set to an orientation of portrait.");
                }

                sheet1.height = pj.pageHeight;
                sheet1.width = pj.pageWidth;
                sheet2.height = pj.pageHeight;
                sheet2.width = pj.pageWidth;

                try
                {
                    pj.addPage(sheet1);
                    pagesToPrint++;
                }
                catch (error:Error)
                {
                // Respond to error.
                }

                try
                {
                    pj.addPage(sheet2);
                    pagesToPrint++;
                }
                catch (error:Error)
                {
                    // Respond to error.
                }

                if (pagesToPrint > 0)
                {
                    pj.send();
                }
            }
        }
    }
}
```

Exemple d'impression : mise à l'échelle, recadrage et ajustement

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans certains cas, il s'avère utile d'ajuster la taille (ou d'autres propriétés) d'un objet d'affichage à imprimer afin de tenir compte des différences entre son aspect à l'écran et le rendu sur papier. Lorsque vous modifiez les propriétés d'un objet d'affichage avant l'impression (par exemple à l'aide des propriétés `scaleX` et `scaleY`), n'oubliez pas que si l'objet redimensionné reste plus grand que le rectangle défini comme zone d'impression, il est recadré. Il est également judicieux d'envisager de rétablir les propriétés après l'impression des pages.

Le code suivant modifie les dimensions de l'objet d'affichage `txt` (sans altérer la zone d'arrière-plan verte). Pour finir, le champ de texte est recadré en fonction des dimensions du rectangle spécifié. Après l'impression, le champ de texte retrouve sa taille originale d'affichage à l'écran. Si l'utilisateur annule la tâche d'impression dans la boîte de dialogue d'impression du système d'exploitation, le contenu du moteur d'exécution de Flash est modifié pour avertir l'utilisateur de cette annulation.

```
package
{
    import flash.printing.PrintJob;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.display.Stage;
    import flash.geom.Rectangle;

    public class PrintScaleExample extends Sprite
    {
        private var bg:Sprite;
        private var txt:TextField;

        public function PrintScaleExample():void
        {
            init();
            draw();
            printPage();
        }

        private function printPage():void
        {
            var pj:PrintJob = new PrintJob();
            txt.scaleX = 3;
            txt.scaleY = 2;
            if (pj.start())
            {
                trace(">> pj.orientation: " + pj.orientation);
                trace(">> pj.pageWidth: " + pj.pageWidth);
                trace(">> pj.pageHeight: " + pj.pageHeight);
                trace(">> pj.paperWidth: " + pj.paperWidth);
                trace(">> pj.paperHeight: " + pj.paperHeight);

                try
                {
                    pj.addPage(this, new Rectangle(0, 0, 100, 100));
                }
                catch (error:Error)
                {
                    // Do nothing.
                }
                pj.send();
            }
            else
            {
                txt.text = "Print job canceled";
            }
            // Reset the txt scale properties.
            txt.scaleX = 1;
            txt.scaleY = 1;
```

```
        }

        private function init():void
        {
            bg = new Sprite();
            bg.graphics.beginFill(0x00FF00);
            bg.graphics.drawRect(0, 0, 100, 200);
            bg.graphics.endFill();

            txt = new TextField();
            txt.border = true;
            txt.text = "Hello World";
        }

        private function draw():void
        {
            addChild(bg);
            addChild(txt);
            txt.x = 50;
            txt.y = 50;
        }
    }
}
```

Exemple d'impression : options d'impression et de mise en page

Adobe AIR 2 et ultérieur

L'exemple suivant initialise les paramètres PrintJob relatifs au nombre d'exemplaires, au format du papier (legal) et à l'orientation de la page (paysage). Il impose l'affichage de la boîte de dialogue de mise en page avant de démarrer la tâche d'impression en affichant la boîte de dialogue d'impression.

```
package
{
    import flash.printing.PrintJob;
    import flash.printing.PrintJobOrientation;
    import flash.printing.PaperSize;
    import flash.printing.PrintUIOptions;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.display.Stage;
    import flash.geom.Rectangle;

    public class PrintAdvancedExample extends Sprite
    {
        private var bg:Sprite = new Sprite();
        private var txt:TextField = new TextField();
        private var pj:PrintJob = new PrintJob();
        private var uiOpt:PrintUIOptions = new PrintUIOptions();

        public function PrintAdvancedExample():void
        {
```

```
        initPrintJob();
        initContent();
        draw();
        printPage();
    }

    private function printPage():void
    {
        //test for dialog support as a static property of PrintJob class
        if (PrintJob.supportsPageSetupDialog)
        {
            pj.showPageSetupDialog();
        }
        if (pj.start2(uiOpt, true))
        {
            try
            {
                pj.addPage(this, new Rectangle(0, 0, 100, 100));
            }
            catch (error:Error)
            {
                // Do nothing.
            }
            pj.send();
        }
        else
        {
            txt.text = "Print job terminated";
            pj.terminate();
        }
    }

    private function initContent():void
    {
        bg.graphics.beginFill(0x00FF00);
        bg.graphics.drawRect(0, 0, 100, 200);
        bg.graphics.endFill();

        txt.border = true;
```

```
            txt.text = "Hello World";
        }

        private function initPrintJob():void
        {
            pj.selectPaperSize(PaperSize.LEGAL);
            pj.orientation = PrintJobOrientation.LANDSCAPE;
            pj.copies = 2;
            pj.jobName = "Flash test print";
        }

        private function draw():void
        {
            addChild(bg);
            addChild(txt);
            txt.x = 50;
            txt.y = 50;
        }
    }
}
```

Chapitre 55 : Geolocation

Si un périphérique prend en charge la géolocalisation, vous disposez de l'API de géolocalisation pour connaître l'emplacement géographique actuel du périphérique. Si le périphérique prend en charge cette fonction, vous pouvez obtenir des informations de géolocalisation. Parmi ces informations figurent l'altitude, la précision, la direction, la vitesse et l'horodatage de la dernière modification de l'emplacement.

La classe Geolocation distribue des événements `update` en réponse au capteur d'emplacement du périphérique. Un événement `update` est un objet GeolocationEvent.

Voir aussi

flash.sensors.Geolocation

flash.events.GeolocationEvent

Détection des changements de géolocalisation

Pour utiliser le capteur de géolocalisation, instanciez un objet Geolocation et enregistrez-vous pour recevoir les événements `update` qu'il distribue. Un événement `update` est un objet GeolocationEvent. Il possède huit propriétés :

- `altitude` : renvoie l'altitude, en mètres.
- `heading` : renvoie la direction du mouvement (par rapport au nord géographique), en degrés.
- `horizontalAccuracy` : renvoie la précision horizontale, en mètres.
- `latitude` : renvoie la latitude, en degrés.
- `longitude` : renvoie la longitude, en degrés.
- `speed` : renvoie la vitesse en mètres par seconde.
- `timestamp` : nombre de millisecondes au moment où se produit l'événement après l'initialisation du moteur d'exécution.
- `verticalAccuracy` : renvoie la précision verticale, en mètres.

La propriété `timestamp` est un objet int. Les autres propriétés sont des objets Number.

Exemple de base qui affiche les données de géolocalisation dans un champ de texte :

```
var geo:Geolocation;
if (Geolocation.isSupported)
{
    geo = new Geolocation();
    geo.addEventListener(GeolocationEvent.UPDATE, updateHandler);
}
else
{
    geoTextField.text = "Geolocation feature not supported";
}
function updateHandler(event:GeolocationEvent):void
{
    geoTextField.text = "latitude: " + event.latitude.toString() + "\n"
            + "longitude: " + event.longitude.toString() + "\n"
            + "altitude: " + event.altitude.toString()
            + "speed: " + event.speed.toString()
            + "heading: " + event.heading.toString()
            + "horizontal accuracy: " + event.horizontalAccuracy.toString()
            + "vertical accuracy: " + event.verticalAccuracy.toString()
}
```

Pour exploiter cet exemple, veillez à créer le champ de texte `geoTextField` et à l'ajouter à la liste d'affichage avant d'utiliser le code.

Vous pouvez ajuster l'intervalle de temps qui sépare les événements de géolocalisation en appelant la méthode `setRequestedUpdateInterval()` de l'objet Geolocation. Cette méthode ne gère qu'un seul paramètre, `interval`, qui correspond à la fréquence de mise à jour requise, en millisecondes :

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

L'intervalle réel entre les mises à jour de la géolocalisation peut être supérieur ou inférieur à cette valeur. Toute modification de l'intervalle de mise à jour affecte l'ensemble des écouteurs enregistrés. Si vous n'appelez pas la méthode `setRequestedUpdateInterval()`, l'application reçoit des mises à jour à la fréquence définie par défaut sur le périphérique.

L'utilisateur peut interdire à une application d'accéder aux données de géolocalisation. L'iPhone avertit par exemple l'utilisateur lorsqu'une application tente d'obtenir des données de géolocalisation. En réponse à l'invite, l'utilisateur peut refuser à l'application tout accès aux données de géolocalisation. L'objet Geolocation distribue un événement `status` lorsque l'utilisateur refuse l'accès aux données de géolocalisation. Il possède également une propriété `muted`, définie sur `true` lorsque le capteur de géolocalisation n'est pas disponible. L'objet Geolocation distribue un événement `status` lorsque la propriété `muted` change. Le code suivant indique comment détecter que les données de géolocalisation ne sont pas disponibles :

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.GeolocationEvent;
    import flash.events.MouseEvent;
    import flash.events.StatusEvent;
    import flash.sensors.Geolocation;
    import flash.text.TextField;
    import flash.text.TextFormat;

    public class GeolocationTest extends Sprite
    {

        private var geo:Geolocation;
        private var log:TextField;

        public function GeolocationTest()
        {
            super();
            stage.align = StageAlign.TOP_LEFT;
            stage.scaleMode = StageScaleMode.NO_SCALE;
            setUpTextField();

            if (Geolocation.isSupported)
            {
                geo = new Geolocation();
                if (!geo.muted)
                {
                    geo.addEventListener(GeolocationEvent.UPDATE, geoUpdateHandler);
                }
                geo.addEventListener(StatusEvent.STATUS, geoStatusHandler);
            }
            else
            {
                log.text = "Geolocation not supported";
            }
        }

        public function geoUpdateHandler(event:GeolocationEvent):void
        {
            log.text = "latitude : " + event.latitude.toString() + "\n";
            log.appendText("longitude : " + event.longitude.toString() + "\n");
        }

        public function geoStatusHandler(event:StatusEvent):void
        {
            if (geo.muted)
                geo.removeEventListener(GeolocationEvent.UPDATE, geoUpdateHandler);
            else
                geo.addEventListener(GeolocationEvent.UPDATE, geoStatusHandler);
        }

        private function setUpTextField():void
```

```
        {
            log = new TextField();
            var format:TextFormat = new TextFormat("_sans", 24);
            log.defaultTextFormat = format;
            log.border = true;
            log.wordWrap = true;
            log.multiline = true;
            log.x = 10;
            log.y = 10;
            log.height = stage.stageHeight - 20;
            log.width = stage.stageWidth - 20;
            log.addEventListener(MouseEvent.CLICK, clearLog);
            addChild(log);
        }
        private function clearLog(event:MouseEvent):void
        {
            log.text = "";
        }
    }
}
```

Remarque : *les iPhone de première génération, qui n'intègrent pas d'unité GPS, ne distribuent que rarement des événements* `update`*. Sur ces périphériques, un objet Geolocation distribue initialement un ou deux événements* `update`*. Il distribue ensuite des événements* `update` *lorsque les informations sont sensiblement modifiées.*

Vérification de la prise en charge de la géolocalisation

La propriété `Geolocation.isSupported` permet de vérifier si l'environnement d'exécution prend en charge cette fonction :

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

A l'heure actuelle, la géolocalisation n'est prise en charge que par les applications ActionScript pour l'iPhone et dans Flash Lite 4. Si `Geolocation.isSupported` correspond à `true` à l'exécution, la géolocalisation est prise en charge.

Certains modèles d'iPhone ne sont pas équipés d'une unité GPS. Ces modèles font appel à d'autres techniques, telles que la localisation par triangulation des téléphones portables, pour obtenir les données de géolocalisation. Pour ces modèles, ou sur tout iPhone dont l'unité GPS est désactivée, un objet Geolocation risque de ne distribuer qu'un ou deux événements `update` initiaux.

Chapitre 56 : Internationalisation des applications

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Le package flash.globalization simplifie la création de logiciels internationaux qui s'adaptent aux conventions de différentes langues et régions.

Voir aussi

Package flash.globalization

« Localisation d'applications » à la page 996

Charles Bihis : Want to Localize your Flex/AIR Apps? (disponible en anglais uniquement)

Principes de base de l'internationalisation des applications

Les termes globalisation et internationalisation sont parfois considérés comme interchangeables. La plupart des définitions de ces termes indiquent toutefois que globalisation se réfère à une combinaison de processus commerciaux et d'ingénierie, tandis qu'internationalisation relève exclusivement de l'ingénierie.

Quelques termes importants sont définis ci-dessous :

Globalisation Large éventail de processus commerciaux et d'ingénierie destinés à assurer la préparation et le lancement globaux de produits et d'actions d'entreprise. La globalisation rassemble des activités d'ingénierie telles que l'internationalisation, la localisation et la différentiation culturelle, ainsi que des activités commerciales telles que la gestion de produits, la planification financière, le marketing et les tâches d'ordre juridiques. La globalisation est parfois abrégée sous la forme *G11n* (G, puis 11 autres lettres, puis la lettre n). *« C'est aux entreprises qu'incombe la globalisation. »*

Internationalisation Processus d'ingénierie destiné à « dé-régionaliser » un produit pour assurer la prise en charge de divers scripts, langues et conventions culturelles (telles que les devises, les règles de tri, les formats numériques et de date, etc.) sans avoir à modifier la conception du produit ou le recompiler. Ce processus peut être divisé en deux jeux d'activités, prise en charge et localisation. Parfois abrégée sous la forme *I18n*, l'internationalisation est également désignée par l'expression anglaise *world-readiness*. *« C'est aux ingénieurs qu'incombe l'internationalisation. »*

Localisation Processus consistant à adapter un produit ou un service à une langue et une culture données, ainsi qu'à une apparence locale appropriée. La localisation est parfois abrégée sous la forme *L10n*. *« C'est aux traducteurs qu'incombe la localisation. »*

Différentiation culturelle Processus relevant de l'ingénierie destiné à développer ou adapter des fonctionnalités déterminées en vue de répondre aux besoins uniques d'une culture. Parmi les exemples disponibles figurent notamment les fonctionnalités de publication japonaises intégrées à Adobe InDesign et la prise en charge de Hanko dans Adobe Acrobat.

Quelques autres termes importants associés à l'internationalisation sont définis ci-dessous :

Jeu de caractères Caractères utilisés par une langue ou un groupe de langues. Un jeu de caractères comprend les caractères nationaux, les caractères spéciaux (tels les signes de ponctuation et les symboles mathématiques), les chiffres et les caractères de contrôle informatiques.

Classement Tri du texte dans un ordre adapté à des paramètres régionaux déterminés.

Paramètres régionaux Valeur qui représente la langue et les conventions culturelles en vigueur dans une région géographique, politique ou culturelle (soit, dans de nombreux cas, un pays unique). A cette valeur correspond un identifiant de paramètres régionaux (ID de paramètres régionaux) unique. L'ID de paramètres régionaux permet de rechercher un jeu de données régionales qui assure une prise en charge régionale appropriée. Cette prise en charge s'applique aux unités de mesure, à l'analyse et au formatage des nombres et des dates, etc.

Regroupement de ressources Jeu stocké d'éléments propres aux paramètres régionaux utilisés par une application. Un regroupement de ressources contient généralement tous les éléments de texte de l'interface utilisateur de l'application. Au sein du regroupement, ces éléments sont traduits dans la langue correspondant aux paramètres régionaux. Le regroupement de ressources comporte parfois d'autres paramètres qui modifient la mise en forme ou le comportement de l'interface utilisateur pour des paramètres régionaux déterminés. Il est ainsi susceptible de contenir d'autres types de médias ou des références à ces derniers, propres aux paramètres régionaux.

Présentation du package flash.globalization

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Le package flash.globalization exploite les fonctionnalités de prise en charge culturelle du système d'exploitation sous-jacent. Il simplifie la programmation d'applications conformes aux conventions culturelles de chaque utilisateur.

Parmi les principales classes du package figurent :

- La classe Collator, qui gère le tri et la concordance des chaînes.
- La classe CurrencyFormatter, qui formate les nombres convertis en chaînes de montant de devise et analyse les symboles et montants de devise extraits de chaînes d'entrée.
- La classe DateTimeFormatter, qui formate les valeurs de date.
- La classe LocaleID, qui extrait les informations relatives à des paramètres régionaux donnés.
- La classe NumberFormatter, qui formate et analyse les valeurs numériques.
- La classe StringTools, qui gère la conversion de la casse des chaînes gérées par les paramètres régionaux.

Package flash.globalization et localisation des ressources

Le package flash.globalization ne gère pas la localisation des ressources. Vous pouvez toutefois utiliser l'ID de paramètres régionaux flash.globalization en tant que principale valeur d'extraction des ressources localisées par le biais d'autres techniques. Il est ainsi possible de localiser les ressources d'une application créées dans Flex à l'aide des classes ResourceManager et ResourceBundle. Pour plus d'informations, voir Localisation d'applications Flex.

Adobe AIR 1.1 contient également quelques fonctions destinées à faciliter la localisation des applications AIR, comme indiqué à la section « Localisation d'applications AIR » à la page 997.

Guide général de l'internationalisation d'une application

La procédure ci-dessous décrit une approche standard de haut niveau de l'internationalisation d'une application à l'aide du package flash.globalization :

1. Déterminez ou définissez les paramètres régionaux.
2. Créez une occurrence d'une classe de service (Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter ou StringTools).
3. Recherchez les erreurs et les paramètres de substitution par le biais des propriétés lastOperationStatus.
4. Formatez et affichez les informations à l'aide des paramètres régionaux en vigueur.

L'étape suivante consiste à charger et afficher les chaînes et ressources de l'interface utilisateur propres aux paramètres régionaux. Cette étape est susceptible d'inclure des tâches telles que :

- redimensionner l'interface utilisateur en fonction des longueurs de chaîne par le biais des fonctions de mise en forme automatique ;
- sélectionner les polices appropriées et les polices de substitution prises en charge ;
- prendre en charge d'autres systèmes d'écriture à l'aide de FTE (Flash Text Engine) ;
- vérifier la gestion appropriée des éditeurs de méthode d'entrée.

Recherche des erreurs et paramètres de substitution

Les classes de service flash.globalization adoptent toutes une approche similaire pour identifier les erreurs. En cas de non-disponibilité des paramètres régionaux requis, elles ont également toutes recours aux paramètres régionaux pris en charge par le système d'exploitation de l'utilisateur.

L'exemple suivant indique comment rechercher les erreurs et paramètres de substitution lors de l'instanciation de classes de service. Chaque classe de service possède une propriété lastOperationStatus, qui indique si la dernière méthode utilisée a déclenché des messages d'erreur ou des avertissements.

```
var nf:NumberFormatter = new NumberFormatter("de-DE");
if(nf.lastOperationStatus != LastOperationStatus.NO_ERROR)
{
    if(nf.lastOperationStatus == LastOperationStatus.USING_FALLBACK_WARNING)
    {
        // perform fallback logic here, if needed
        trace("Warning - Fallback locale ID: " + nf.actualLocaleIDName);
    }
    else
    {
        // perform error handling logic here, if needed
        trace("Error: " + nf.lastOperationStatus);
    }
}
```

Cet exemple se contente de suivre un message si un ID de paramètres régionaux de substitution est utilisé ou s'il se produit une erreur. Le cas échéant, l'application peut mettre en œuvre d'autres opérations logiques de gestion des erreurs. Vous pouvez, par exemple, afficher un message à l'intention de l'utilisateur ou imposer à l'application d'utiliser des paramètres régionaux spécifiques et pris en charge.

Identification des paramètres régionaux

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Les paramètres régionaux identifient une combinaison donnée de conventions linguistiques et culturelles en vigueur dans un pays ou une région.

Un identifiant de paramètres régionaux peut être géré en toute sécurité en tant que chaîne. Vous disposez toutefois de la classe LocaleID pour obtenir d'autres informations relatives aux paramètres régionaux.

Procédez comme suit pour créer un objet LocaleID :

```
var locale:LocaleID = new LocaleID("es-MX");
```

Une fois l'objet LocaleID créé, vous pouvez extraire des données relatives à l'ID de paramètres régionaux. Faites appel aux méthodes `getKeysAndValues()`, `getLanguage()`, `getRegion()`, `getScript()`, `getVariant()` et `isRightToLeft()`, ainsi qu'à la propriété `name`.

Les valeurs extraites de ces méthodes et propriétés contiennent parfois des informations complémentaires sur les paramètres régionaux qu'il est impossible de récupérer directement de l'identifiant de paramètres régionaux.

Lorsqu'une application crée un service qui gère les paramètres régionaux, telle une fonctionnalité de formatage de dates, elle doit stipuler les paramètres régionaux prévus. Etant donné que la liste des paramètres régionaux pris en charge varie d'un système d'exploitation à l'autre, il arrive parfois que les paramètres régionaux requis ne soient pas disponibles.

Flash Player essaie d'abord de trouver le code de langue des paramètres régionaux requis. Il tente ensuite de préciser les paramètres régionaux en trouvant un système d'écriture (script) et une région correspondants. Exemple :

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

Dans cet exemple, le constructeur `LocaleID()` a extrait les données relatives aux paramètres régionaux qui correspondent de plus près au code de langue « es » associé à l'utilisateur.

Définition de l'identifiant de paramètres régionaux

Vous disposez de plusieurs méthodes pour définir les paramètres régionaux actifs d'une application :

- Codez en dur un ID de paramètres régionaux unique dans l'application. Cette approche est courante, mais ne prend pas en charge l'internationalisation de l'application.
- Utilisez les préférences d'ID de paramètres régionaux extraites du système d'exploitation ou du navigateur de l'utilisateur, voire d'autres préférences utilisateur. Cette technique produit généralement les paramètres régionaux les plus adaptés à l'utilisateur, mais manque parfois de précision. Les paramètres du système d'exploitation risquent en effet de ne pas refléter les préférences réelles de l'utilisateur. L'utilisateur pourrait, par exemple, partager un ordinateur avec un collègue et être incapable de modifier les paramètres régionaux préférés du système d'exploitation.
- Après avoir défini l'ID des paramètres régionaux en fonction des préférences de l'utilisateur, autorisez ce dernier à sélectionner les paramètres régionaux dans une liste d'options prises en charge. Cette stratégie est généralement recommandée si l'application prend en charge plusieurs jeux de paramètres régionaux.

Vous pouvez faire appel à la troisième option, comme suit :

1. Récupérez la liste des paramètres régionaux ou langues préférés de l'utilisateur à partir d'un profil utilisateur, des paramètres du navigateur, des paramètres du système d'exploitation ou d'un cookie. (L'application doit implémenter elle-même cette logique. La bibliothèque flash.globalization ne prend en effet pas directement en charge ces préférences.)
2. Identifiez les paramètres régionaux pris en charge par l'application et sélectionnez la valeur la plus adaptée par défaut. Faites appel à la méthode LocaleID.determinePreferredLocales() pour trouver les paramètres régionaux adaptés à un utilisateur en fonction de ses paramètres régionaux préférés et des paramètres régionaux pris en charge par le système d'exploitation.
3. Autorisez l'utilisateur à modifier les paramètres régionaux par défaut s'ils ne lui conviennent pas.

Restrictions associées aux autres classes de paramètres régionaux et de langue

La classe `fl.lang.Locale` permet de remplacer les chaînes de texte basées sur des paramètres régionaux en utilisant les regroupements de ressources qui contiennent les valeurs de chaîne. Cette classe ne prend toutefois pas en charge les autres fonctions d'internationalisation telles que le formatage, le tri et la mise en correspondance des nombres, des devises ou de la date. Elle est par ailleurs réservée à Flash Professional.

Vous pouvez également récupérer le code de langue actif associé au système d'exploitation à l'aide de la propriété `flash.system.Capabilities.language`. Cette propriété n'extrait toutefois que le code de langue ISO 639-1 à deux caractères, plutôt que l'ID de paramètres régionaux complet, et ne prend en charge qu'un jeu déterminé de paramètres régionaux.

Dans AIR 1.5, vous disposez de la propriété `flash.system.Capabilities.languages`. Cette propriété génère un tableau contenant les langues d'interface privilégiées par l'utilisateur. Elle n'est donc pas sujette aux restrictions de `Capabilities.language`.

Formatage des nombres

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Le format d'affichage des valeurs numériques varie considérablement d'une région à l'autre. Le nombre 123456,78 est par exemple formaté comme suit selon les paramètres régionaux en vigueur :

Paramètres régionaux	Formatage du nombre
en-US (anglais, Etats-Unis)	-123,456.78
de-DE (allemand, Allemagne)	-123.456,78
fr-FR (français, France)	-123 456,78
de-CH (allemand, Suisse)	-123'456.78
en-IN (anglais, Inde)	-1,23,456.78
La plupart des paramètres régionaux arabes	123,456.78-

Un grand nombre de facteurs affectent les formats numériques, tels que :

- Séparateurs : le séparateur décimal est placé entre le nombre entier et la partie fractionnelle d'un nombre. Il peut correspondre à un point, une virgule ou un autre caractère. Le séparateur de milliers peut être un point, une virgule, un espace insécable ou un autre caractère.
- Types de regroupement : le nombre de chiffres entre chaque séparateur de milliers à gauche du séparateur décimal peut correspondre à deux, trois, voire une autre valeur.
- Indicateurs de nombre négatif : signe moins inséré sur la gauche ou la droite du nombre négatif, ou nombre négatif placé entre parenthèses pour les applications financières. Ainsi, le nombre 19 négatif peut être affiché comme suit : -19, 19- ou (19).
- Zéros à droite ou à gauche : certaines conventions culturelles ajoutent des zéros à gauche ou à droite des nombres affichés. Ainsi, la valeur 0,17 peut être affichée comme suit : .17, 0,17 ou 0,170, etc.
- Jeux de caractères numériques : de nombreuses langues, parmi lesquelles l'arabe, l'hindi et le japonais, utilisent différents jeux de caractères numériques. Le package flash.globalization prend en charge tous les jeux de caractères numériques mappés sur les chiffres 0 à 9.

La classe NumberFormatter prend en compte tous ces facteurs lors du formatage de valeurs numériques.

Utilisation de la classe NumberFormatter

La classe NumberFormatter formate les valeurs numériques (de type int, uint ou Number) en fonction des conventions de paramètres régionaux donnés.

L'exemple suivant illustre la façon la plus simple de formater un nombre en fonction des propriétés de formatage par défaut gérées par le système d'exploitation de l'utilisateur :

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

Le résultat varie selon les paramètres régionaux en vigueur et les préférences de l'utilisateur. Ainsi, si les paramètres régionaux de l'utilisateur correspondent à fr-FR, la valeur est formatée comme suit :

-123.456,789

Si vous souhaitez vous limiter au formatage d'un nombre en fonction de paramètres régionaux donnés, quels que soient les paramètres de l'utilisateur, définissez spécifiquement le nom des paramètres régionaux. Exemple :

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

Dans ce cas de figure, le résultat est le suivant :

-123'456.789

La méthode formatNumber() traite la valeur Number comme un paramètre. La classe NumberFormatter gère également une méthode formatInt(), qui gère l'entrée int, et une méthode formatUint(), qui gère l'entrée uint.

Pour contrôler explicitement la logique de formatage, définissez les propriétés de la classe NumberFormatter, comme illustré ci-dessous :

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
nf.negativeNumberFormat = 0;
nf.fractionalDigits = 5;
nf.trailingZeros = true;
nf.decimalSeparator = ",";
nf.useGrouping = false;
trace(nf.formatNumber(-123456.789)); //(123456.78900)
```

Cet exemple commence par créer un objet NumberFormatter, puis :

- définit le format numérique négatif de sorte à utiliser des parenthèses (comme dans les applications financières) ;
- définit le nombre de chiffres après le séparateur décimal sur 5 ;
- stipule l'insertion de zéros à droite pour garantir la présence de cinq décimales ;
- définit le séparateur décimal sur une virgule ;
- indique à la fonctionnalité de formatage de ne pas utiliser de séparateur de milliers.

Remarque : *si certaines de ces propriétés sont modifiées, le format numérique résultant ne correspond plus au format préféré associé aux paramètres régionaux indiqués. Ne faites appel à certaines de ces propriétés que si la prise en charge des paramètres régionaux est superflue, comme dans les cas de figure suivants : si un aspect unique du format doit être contrôlé avec précision (le nombre de zéros à droite, par exemple) ou si l'utilisateur demande directement la modification (via le Panneau de configuration Windows, par exemple).*

Analyse des chaînes contenant des valeurs numériques

La classe NumberFormatter peut également extraire des valeurs numériques de chaînes conformes aux conventions de formatage stipulées par les paramètres régionaux. La méthode NumberFormatter.parseNumber() extrait une valeur numérique unique d'une chaîne. Exemple :

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String =  "-1,234,567.890"
var parsedNumber:Number = nf.parseNumber(inputNumberString);
trace("Value:" + parsedNumber); // -1234567.89
trace("Status:" + nf.lastOperationStatus); // noError
```

La méthode parseNumber() gère les chaînes qui ne contiennent que des caractères de formatage de chiffres et de nombres, tels que les signes moins et les séparateurs. Si la chaîne contient d'autres caractères, un code d'erreur est défini :

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String =  "The value is 1,234,567.890"
var parsedNumber:Number = nf.parseNumber(inputNumberString);
trace("Value:" + parsedNumber); // NaN
trace("Status:" + nf.lastOperationStatus); // parseError
```

Pour extraire des nombres de chaînes qui contiennent également des caractères alphabétiques, faites appel à la méthode NumberFormatter.parse() :

```
var nf:NumberFormatter = new NumberFormatter( "en-US" );
var inputNumberString:String = "The value is 123,456,7.890";
var parseResult:NumberParseResult = nf.parse(inputNumberString);
trace("Value:" + parseResult.value); // 1234567.89
trace("startIndex: " + parseResult.startIndex); // 14
trace("Status:" + nf.lastOperationStatus); // noError
```

La méthode parse() renvoie un objet NumberParseResult, dont la propriété Value contient la valeur numérique analysée. La propriété startIndex indique l'index du premier caractère numérique détecté. Les propriétés startIndex et endIndex permettent d'extraire les sections de chaîne qui précèdent ou suivent les chiffres.

Formatage des valeurs en devise

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Les formats d'affichage des valeurs en devise varient tout autant que les formats numériques. La valeur 123456,78 $ est par exemple formatée comme suit selon les paramètres régionaux en vigueur :

Paramètres régionaux	Formatage du nombre
en-US (anglais, Etats-Unis)	$123,456.78
de-DE (allemand, Allemagne)	123.456,78 $
en-IN (anglais, Inde)	$ 1,23,456.78

Le formatage des valeurs en devise est affecté par les mêmes facteurs que le formatage numérique, auxquels s'ajoutent les facteurs complémentaires suivants :

- Code ISO de devise : code de devise ISO 4217 de trois lettres des paramètres régionaux actuellement spécifiés (USD ou EUR, par exemple).
- Symbole de devise : chaîne ou symbole de devise correspondant aux paramètres régionaux actuellement spécifiés ($ ou €, par exemple).
- Format de devise négatif : définit l'emplacement du symbole de devise et du symbole négatif ou des parenthèses par rapport à la partie numérique du montant en devise.
- Format de devise positif : définit l'emplacement du symbole de devise par rapport à la partie numérique du montant en devise.

Utilisation de la classe CurrencyFormatter

La classe CurrencyFormatter transforme les valeurs numériques en chaînes contenant des chaînes de devise et des nombres formatés, conformément aux conventions de paramètres régionaux donnés.

Lorsque vous instanciez un nouvel objet CurrencyFormatter, il définit la devise sur la devise associée par défaut aux paramètres régionaux en vigueur.

L'exemple suivant indique qu'un objet CurrencyFormatter créé à partir de paramètres régionaux allemands part du principe que les montants en devise sont exprimés en euros :

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Dans la plupart des cas, ne vous fiez pas à la devise associée par défaut aux paramètres régionaux. Si les paramètres régionaux par défaut d'un utilisateur ne sont pas pris en charge, la classe CurrencyFormatter affecte des paramètres régionaux de substitution. Les paramètres régionaux de substitution utilisent parfois une autre devise par défaut. Par ailleurs, les formats en devise doivent généralement sembler corrects à l'utilisateur, même si les montants ne sont pas exprimés dans sa devise locale. Ainsi, un utilisateur canadien peut être amené à afficher les tarifs d'une entreprise allemande en euros, mais formatés dans le style canadien.

La méthode CurrencyFormatter.setCurrency() spécifie la chaîne et le symbole de devise exacts à utiliser.

L'exemple suivant affiche les montants en devise en euros à l'intention des utilisateurs québécois :

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

La méthode setCurrency() permet également de réduire les risques de confusion en définissant des symboles de devise non ambigus. Exemple :

```
cf.setCurrency("USD","US$");
```

Par défaut, la méthode format() affiche un code de devise ISO 4217 de trois caractères au lieu du symbole de devise. Les codes ISO 4217 ne sont pas ambigus et ne nécessitent pas de localisation. Un grand nombre d'utilisateurs préfèrent toutefois les symboles de devise aux codes ISO.

La classe CurrencyFormatter peut vous aider à décider du symbole utilisé par une chaîne en devise formatée : un symbole de devise ($ ou €, par exemple) ou une chaîne de devise ISO de trois caractères (USD ou EUR, par exemple). Ainsi, un montant exprimé en dollars canadiens pourrait être affiché sous la forme $200 à l'intention d'un utilisateur qui réside au Canada. Pour un utilisateur qui réside aux Etats-Unis, il pourrait par contre être affiché sous la forme CAD 200. La méthode formattingWithCurrencySymbolIsSafe() permet de déterminer si le symbole de devise d'un montant risque d'être ambigu ou incorrect en fonction des paramètres régionaux de l'utilisateur.

L'exemple suivant affiche une valeur en euros dans un format adapté aux paramètres régionaux en-US. Selon les paramètres régionaux de l'utilisateur, la chaîne sortie contient soit le code de devise ISO, soit le symbole de devise.

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-CA");

if (cf.formattingWithCurrencySymbolIsSafe("USD"))
{
    trace(cf.format(1234567.89, true)); // $1,234,567.89
}
else
{
    cf.setCurrency("USD", "$");
    trace(cf.format(1234567.89)); // USD1,234,567.89
}
```

Analyse des chaînes contenant des valeurs en devise

La classe CurrencyFormatter peut également extraire un montant et une chaîne de devise d'une chaîne de sortie conforme aux conventions de formatage des paramètres régionaux. La méthode CurrencyFormatter.parse() enregistre le montant et la chaîne de devise analysés dans un objet CurrencyParseResult, comme illustré ci-après :

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-US" );
var inputCurrencyString:String = "(GBP 123,56,7.890)";
var parseResult:CurrencyParseResult = cf.parse(inputCurrencyString);
trace("parsed amount: " + parseResult.value); // -1234567.89
trace("currencyString: " + parseResult.currencyString ); // GBP
```

La partie chaîne de devise de la chaîne d'entrée peut contenir un symbole de devise, un code ISO de devise et des caractères texte. La position de la chaîne de devise, de l'indicateur de nombre négatif et de la valeur numérique correspond aux formats spécifiés par les propriétés negativeCurrencyFormat et positiveCurrencyFormat. Exemple :

```
var cf:CurrencyFormatter = new CurrencyFormatter( "en-US" );
var inputCurrencyString:String = "Total $-123,56,7.890";
var parseResult:CurrencyParseResult = cf.parse(inputCurrencyString);
trace("status: " + cf.lastOperationStatus ); // parseError
trace("parsed amount: " + parseResult.value); // NaN
trace("currencyString: " + parseResult.currencyString ); //
cf.negativeCurrencyFormat = 2;
parseResult = cf.parse(inputCurrencyString);
trace("status: " + cf.lastOperationStatus ); // noError
trace("parsed amount: " + parseResult.value); // -123567.89
trace("currencyString: " + parseResult.currencyString ); // Total $
```

Dans cet exemple, la chaîne d'entrée contient une chaîne de devise, suivie d'un signe moins et d'un nombre. Toutefois, la valeur negativeCurrencyFormat par défaut associée aux paramètres régionaux en-US spécifie que l'indicateur négatif est placé en première position. Par conséquent, la méthode parse() génère une erreur et la valeur analysée est NaN.

Une fois negativeCurrencyFormat défini sur 2, qui stipule que la chaîne de devise est placée en première position, la méthode parse() aboutit.

Formatage des dates et heures

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Le format d'affichage de la date et de l'heure varie également considérablement d'une région à l'autre. Voici par exemple comment le 2 janvier 1962 à 13:01 est affiché dans un format court en fonction de divers paramètres régionaux :

Paramètres régionaux	Format de la date et de l'heure
en-US (anglais, Etats-Unis)	1/2/62 1:01pm
fr-FR (français, France)	2/1/62 13:01
ja-JP (japonais, Japon)	1962/2/1 13:01

Utilisation de la classe DateTimeFormatter

La classe DateTimeFormatter formate les valeurs Date en chaînes de date et d'heure conformément aux conventions de paramètres régionaux donnés.

Le formatage est régi par une chaîne modèle qui contient des séquences de lettres remplacées par des valeurs de date ou d'heure. Par exemple, dans le modèle « yyyy/MM », les caractères « yyyy » sont remplacés par une année à quatre chiffres, suivie du caractère « / » et d'un mois à deux chiffres.

La méthode setDateTimePattern() permet de définir explicitement la chaîne modèle. Il est toutefois préférable d'activer la définition automatique du modèle en fonction des préférences du système d'exploitation et des paramètres régionaux de l'utilisateur. Cette stratégie contribue à assurer un résultat approprié d'un point de vue culturel.

L'élément DateTimeFormatter peut représenter des dates et heures en trois styles standard (LONG, MEDIUM et SHORT) et faire appel à un modèle CUSTOM. Vous pouvez utiliser un style pour la date et un autre pour l'heure. Les modèles associés à chaque style varient sensiblement d'un système d'exploitation à l'autre.

Vous pouvez spécifier les styles lors de la création d'un objet DateTimeFormatter. Si les paramètres de style ne sont pas spécifiés, ils sont définis sur DateTimeStyle.LONG par défaut. Vous pouvez modifier les styles ultérieurement à l'aide de la méthode setDateTimeStyles(), comme indiqué dans l'exemple suivant :

```
var date:Date = new Date(2009, 2, 27, 13, 1);
var dtf:DateTimeFormatter = new DateTimeFormatter("en-US",
    DateTimeStyle.LONG, DateTimeStyle.LONG);

var longDate:String = dtf.format(date);
trace(longDate); // March 27, 2009 1:01:00 PM

dtf.setDateTimeStyles(DateTimeStyle.SHORT, DateTimeStyle.SHORT);
var shortDate:String = dtf.format(date);
trace(shortDate); // 3/27/09 1:01 PM
```

Localisation des noms de mois et de jour

De nombreuses applications font appel à des listes de noms de mois et de jour de la semaine dans les calendriers ou listes déroulantes affichés.

La méthode DateTimeFormatter.getMonthNames() permet de récupérer une liste localisée de noms de mois. Le système d'exploitation détermine si les formes complètes et abrégées sont disponibles. Transmettez la valeur DateTimeNameStyle.FULL pour extraire les noms de mois complet. Transmettez la valeur DateTimeNameStyle.LONG_ABBREVIATION ou DateTimeNameStyle.SHORT_ABBREVIATION pour obtenir des versions abrégées.

Dans certaines langues, un nom de mois change (il est alors mis au génitif) s'il est placé à côté de la valeur jour dans un format de date. Si vous envisagez d'utiliser les noms de mois sans spécifier de jour, transmettez la valeur DateTimeNameContext.STANDALONE à la méthode getMonthNames(). Pour utiliser les noms de mois dans les dates formatées, transmettez toutefois la valeur DateTimeNameContext.FORMAT.

```
var dtf:DateTimeFormatter = new DateTimeFormatter("fr-FR");
var months:Vector.<String> = dtf.getMonthNames(DateTimeNameStyle.FULL,
        DateTimeNameContext.STANDALONE);
trace(months[0]); // janvier
months = dtf.getMonthNames(DateTimeNameStyle.SHORT_ABBREVIATION,
         DateTimeNameContext.STANDALONE);
trace(months[0]); // janv.
```

La méthode DateTimeFormatter.getWeekdayNames() fournit une liste localisée de noms de jour de la semaine. La méthode getWeekdayNames() gère les mêmes paramètres nameStyle et context que la méthode getMonthNames().

```
var dtf:DateTimeFormatter = new DateTimeFormatter("fr-FR");
var weekdays:Vector.<String> = dtf.getWeekdayNames(DateTimeNameStyle.FULL,
           DateTimeNameContext.STANDALONE);
trace(weekdays[0]); // dimanche
weekdays = dtf.getWeekdayNames(DateTimeNameStyle.LONG_ABBREVIATION,
           DateTimeNameContext.STANDALONE);
trace(weekdays[0]); // dim.
```

Par ailleurs, la méthode getFirstWeekday() renvoie la valeur d'index du jour qui représente traditionnellement le début de la semaine dans les paramètres régionaux sélectionnés.

Tri et comparaison des chaînes

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Le processus de disposition des éléments dans l'ordre approprié porte le nom de classement. Les règles de classement varient considérablement selon les paramètres régionaux sélectionnés. Elles varient également si vous triez une liste ou que vous mettez en correspondance des éléments similaires (dans un algorithme de recherche de texte, par exemple).

Lors de l'exécution d'un tri, les différences mineures telles que la casse ou les signes diacritiques tels que les accents jouent souvent un rôle important. Ainsi, la lettre ö (o avec un tréma) est généralement considérée comme étant équivalente à la lettre o simple en français ou en anglais. En suédois, cette lettre suit cependant la lettre z. En français comme dans d'autres langues, la présence d'un caractère accentué dans un mot affecte l'ordre de tri d'une liste.

Lors d'une recherche, il est souvent préférable de ne pas tenir compte de la casse ou des signes diacritiques pour augmenter les chances de détection de correspondances appropriées. Ainsi, la recherche des caractères « cote » dans un document français est susceptible de renvoyer « cote », « côte » et « coté ».

Utilisation de la classe Collator

La classe Collator a pour principales méthodes compare(), qui sert principalement à trier, et equals(), qui permet de mettre en correspondance les valeurs.

L'exemple suivant illustre le comportement des méthodes compare() et equals().

```
var words:Array = new  Array("coté", "côte");

var sorter:Collator = new Collator("fr-FR", CollatorMode.SORTING);
words.sort(sorter.compare);
trace(words); // côte,coté

var matcher:Collator = new Collator("fr-FR", CollatorMode.MATCHING);
if (matcher.equals(words[0], words[1]))
{
    trace(words[0] + " = " + words[1]); // côte = coté
}
```

L'exemple commence par créer un objet Collator en mode SORTING pour les paramètres régionaux français-France. Il trie ensuite deux mots dont l'unique différence réside dans la présence de signes diacritiques. Il indique que le mode de comparaison SORTING établit une distinction entre les caractères accentués et non accentués.

Le tri repose sur la transmission d'une référence à la méthode sort() de l'objet Collator en tant que paramètre de la méthode Array.sort(). Ce type d'utilisation d'un objet Collator pour gérer l'ordre de tri est particulièrement efficace.

L'exemple crée ensuite un objet Collator en mode MATCHING. Lorsque l'objet Collator compare les deux mots, il les considère comme égaux. La comparaison MATCHING n'établit ainsi pas de distinction entre les caractères accentués et non accentués.

Personnalisation du comportement de la classe Collator

Par défaut, la classe Collator fait appel aux règles de comparaison de chaînes extraites du système d'exploitation, en fonction des paramètres régionaux et des préférences système de l'utilisateur. Pour personnaliser le comportement des méthodes compare() et equals(), définissez explicitement diverses propriétés. Le tableau suivant recense les propriétés et leur impact sur les comparaisons :

Propriété Collator	Impact
numericComparison	Détermine si les caractères numériques sont traités comme des nombres ou du texte.
ignoreCase	Détermine si les différences de casse sont ignorées.
ignoreCharacterWidth	Détermine si les formes à pleine chasse et à demi-chasse de certains caractères chinois et japonais sont considérées comme égales.
ignoreDiacritics	Détermine si les chaînes qui contiennent les mêmes caractères de base, mais des accents ou autres signes diacritiques différents sont considérées comme égales.
ignoreKanaType	Détermine si les chaînes dont l'unique différence réside dans le type de caractère kana utilisé sont considérées comme égales.
ignoreSymbols	Détermine si les symboles tels que les espaces, les symboles de devise, les symboles mathématiques et autres sont pris en compte.

Le code suivant indique que définir la propriété ignoreDiacritics sur true modifie l'ordre de tri d'une liste de mots français :

```
var words:Array = new  Array("COTE", "coté", "côte", "Coté","cote");
var sorter:Collator = new Collator("fr-CA", CollatorMode.SORTING);
words.sort(sorter.compare);
trace(words); // cote,COTE,côte,coté,Coté

sorter.ignoreDiacritics = true;
words.sort(sorter.compare);
trace(words); // côte,coté,cote,Coté,COTE
```

Conversion de la casse

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

Les langues gèrent également différemment les règles de conversion des majuscules et des minuscules.

Ainsi, dans la plupart des langues basées sur l'alphabet latin, la forme minuscule de la majuscule « I » est «i ». Toutefois, certaines langues, telles que le turc et l'azéri, comportent une autre lettre « ı » sans point. Par conséquent, dans ces langues, un « ı » minuscule sans point se transforme en « I » majuscule. Un « i » minuscule se transforme quant à lui en « İ » majuscule avec un point.

La classe StringTools propose des méthodes qui font appel à des règles propres à chaque langue pour exécuter ces transformations.

Utilisation de la classe StringTools

La classe StringTools contient deux méthodes de transformation de la casse, toLowerCase() et toUpperCase(). Vous créez un objet StringTools en appelant le constructeur à l'aide d'un ID de paramètres régionaux. La classe StringTools extrait les règles de conversion de casse associées aux paramètres régionaux (ou à des paramètres régionaux de substitution) du système d'exploitation. Il est impossible de personnaliser plus encore l'algorithme de conversion de casse.

L'exemple suivant fait appel aux méthodes toUpperCase() et toLowerCase() pour transformer une expression allemande qui comprend la lettre « ß » (Eszett).

```
var phrase:String = "Schloß Neuschwanstein";
var converter:StringTools = new StringTools("de-DE");

var upperPhrase:String = converter.toUpperCase(phrase);
trace(upperPhrase); // SCHLOSS NEUSCHWANSTEIN

var lowerPhrase:String = converter.toLowerCase(upperPhrase);
trace(lowerPhrase);// schloss neuschwanstein
```

La méthode toUpperCase() transforme le « ß » minuscule en « SS » majuscules. Cette transformation ne fonctionne que dans un sens. Lorsque les lettres « SS » sont reconverties en minuscules, le résultat est « ss » et non « ß ».

Exemple : Internationalisation d'une application de suivi des stocks

Flash Player 10.1 et les versions ultérieures, Adobe AIR 2.0 et les versions ultérieures

L'application Global Stock Ticker extrait et affiche des données boursières fictives dans trois Bourses distinctes : Etats-Unis, Japon et Europe. Elle formate les données en fonction des conventions de divers paramètres régionaux.

Cet exemple illustre les fonctions suivantes du package flash.globalization :

- Formatage des nombres en fonction des paramètres régionaux
- Formatage des devises en fonction des paramètres régionaux
- Définition des codes ISO et des symboles de devise
- Formatage de date en fonction des paramètres régionaux
- Extraction et affichage des abréviations de noms de mois appropriées

Pour obtenir les fichiers d'application de cet exemple, voir www.adobe.com/go/learn_programmingAS3samples_flash_fr. Les fichiers d'application de Global Stock Ticker résident dans le dossier Samples/GlobalStockTicker. L'application se compose des fichiers suivants :

Fichier	Description
GlobalStockTicker.mxml ou GlobalStockTicker.fla	Interface utilisateur de l'application pour Flex (MXML) ou Flash (FLA).
styles.css	Styles associés à l'interface utilisateur de l'application (Flex uniquement)
com/example/programmingas3/stockticker/flex/FinGraph.mxml	Composant MXML qui affiche les données boursières simulées sous forme graphique (Flex uniquement)
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	Classe Document contenant la logique de l'interface utilisateur de l'application (Flash uniquement)
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Fonctionnalité de rendu de cellule personnalisé associée au composant DataGrid de Flash (Flash uniquement)

Fichier	Description
com/example/programmingas3/stockticker/FinancialGraph.as	Classe ActionScript qui illustre les données boursières simulées sous forme graphique
com/example/programmingas3/stockticker/Localizer.as	Classe ActionScript qui gère les paramètres régionaux et les devises et traite le formatage localisé des nombres, montants en devise et dates
com/example/programmingas3/stockticker/StockDataModel.as	Classe ActionScript qui stocke toutes les exemples de données de l'application Global Stock Ticker

Présentation de l'interface utilisateur et des exemples de données

Les principaux éléments de l'interface utilisateur de l'application sont les suivants :

- Liste déroulante permettant de sélectionner les paramètres régionaux
- Liste déroulante permettant de sélectionner une Bourse
- Grille contenant les données de six sociétés par Bourse
- Graphique illustrant les données historiques simulées des actions de la société sélectionnée

L'application stocke tous les exemples de données relatifs aux paramètres régionaux, aux Bourses et aux actions des sociétés dans la classe StockDataModel. Une application réelle extrairait les données d'un serveur et les stockerait dans une classe telle que StockDataModel. Dans cet exemple, toutes les données sont codées en dur dans la classe StockDataModel.

Remarque : *les données affichées dans le graphique financier ne correspondent pas nécessairement pas aux données du contrôle grille de données. Le graphique est actualisé à chaque fois qu'une autre société est sélectionnée. Il est proposé à titre d'illustration uniquement.*

Définition des paramètres régionaux

Au terme de la phase de configuration initiale, l'application appelle la méthode Localizer.setLocale() pour créer des objets de formatage associés aux paramètres régionaux par défaut. La méthode setLocale() est également appelée à chaque fois que l'utilisateur sélectionne une nouvelle valeur dans la liste déroulante de paramètres régionaux.

```
public function setLocale(newLocale:String):void
{
    locale  = new LocaleID(newLocale);

    nf = new NumberFormatter(locale.name);
    traceError(nf.lastOperationStatus, "NumberFormatter", nf.actualLocaleIDName);

    cf = new CurrencyFormatter(locale.name);
    traceError(cf.lastOperationStatus, "CurrencyFormatter", cf.actualLocaleIDName);
    symbolIsSafe = cf.formattingWithCurrencySymbolIsSafe(currentCurrency);
    cf.setCurrency(currentCurrency, currentSymbol);
    cf.fractionalDigits = currentFraction;

    df = new DateTimeFormatter(locale.name, DateTimeStyle.LONG, DateTimeStyle.SHORT);
    traceError(df.lastOperationStatus, "DateTimeFormatter", df.actualLocaleIDName);
    monthNames = df.getMonthNames(DateTimeNameStyle.LONG_ABBREVIATION);
}

public function traceError(status:String, serviceName:String, localeID:String) :void
{
    if(status != LastOperationStatus.NO_ERROR)
    {
        if(status == LastOperationStatus.USING_FALLBACK_WARNING)
        {
            trace("Warning - Fallback locale ID used by "
                    + serviceName + ": " + localeID);
        }
        else if (status == LastOperationStatus.UNSUPPORTED_ERROR)
        {
            trace("Error in " + serviceName + ": " + status);
            //abort application
            throw(new Error("Fatal error", 0));
        }
        else
        {
            trace("Error in " + serviceName + ": " + status);
        }
    }
    else
    {
        trace(serviceName + " created for locale ID: " + localeID);
    }
}
```

La méthode setLocale() commence par créer un objet LocaleID. Cet objet simplifie l'obtention ultérieure d'informations détaillées sur les paramètres régionaux en tant que tels, le cas échéant.

La méthode crée ensuite des objets NumberFormatter, CurrencyFormatter et DateTimeFormatter associés aux paramètres régionaux. Au terme de la création de chaque objet de formatage, elle appelle la méthode traceError(). Cette méthode affiche des messages d'erreur et d'avertissement sur la console en cas de problème au niveau des paramètres régionaux requis. (Une application réelle devrait réagir suite à des erreurs de ce type au lieu de se contenter de les suivre.)

Une fois l'objet CurrencyFormatter créé, la méthode setLocale() définit le code ISO de la devise, le symbole de la devise et les propriétés fractionalDigits de la fonctionnalité de formatage sur les valeurs précédemment déterminées. (Ces valeurs sont définies à chaque fois que l'utilisateur sélectionne une nouvelle option dans la liste déroulante de marchés.)

Une fois l'objet DateTimeFormatter créé, la méthode setLocale() extrait également un tableau d'abréviations de noms de mois localisés.

Formatage des données

Les données boursières formatées sont affichées dans un contrôle de type grille de données. Chaque colonne de la grille de données appelle une fonction d'étiquette qui formate la valeur de la colonne par le biais de l'objet de formatage approprié.

Dans la version Flash, par exemple, le code suivant configure les colonnes de la grille de données :

```
var col1:DataGridColumn = new DataGridColumn("ticker");
col1.headerText = "Company";
col1.sortOptions = Array.NUMERIC;
col1.width = 200;

var col2:DataGridColumn = new DataGridColumn("volume");
col2.headerText = "Volume";
col2.width = 120;
col2.cellRenderer = RightAlignedCell;
col2.labelFunction = displayVolume;

var col3:DataGridColumn = new DataGridColumn("price");
col3.headerText = "Price";
col3.width = 70;
col3.cellRenderer = RightAlignedCell;
col3.labelFunction = displayPrice;

var col4:DataGridColumn = new DataGridColumn("change");
col4.headerText = "Change";
col4.width = 120;
col4.cellRenderer = RightAlignedCell;
col4.labelFunction = displayPercent;
```

La version Flex de l'exemple déclare sa grille de données en MXML. Elle définit également des fonctions d'étiquette similaires pour chaque colonne.

Les propriétés labelFunction font référence aux fonctions suivantes, qui appellent les méthodes de formatage de la classe Localizer :

```
private function displayVolume(item:Object):String
{
    return localizer.formatNumber(item.volume, 0);
}

private function displayPercent(item:Object):String
{
    return localizer.formatPercent(item.change )  ;
}

  private function displayPrice(item:Object):String
{
    return localizer.formatCurrency(item.price);
}
```

Les méthodes Localizer configurent et appellent ensuite les fonctionnalités de formatage appropriées :

```
public function formatNumber(value:Number, fractionalDigits:int = 2):String
{
    nf.fractionalDigits = fractionalDigits;
    return nf.formatNumber(value);
}

public function formatPercent(value:Number, fractionalDigits:int = 2):String
{
    // HACK WARNING: The position of the percent sign, and whether a space belongs
    // between it and the number, are locale-sensitive decisions. For example,
    // in Turkish the positive format is %12 and the negative format is -%12.
    // Like most operating systems, flash.globalization classes do not currently
    // provide an API for percentage formatting.
    nf.fractionalDigits = fractionalDigits;
    return nf.formatNumber(value) + "%";
}

public function formatCurrency(value:Number):String
{
    return cf.format(value, symbolIsSafe);
}

public function formatDate(dateValue:Date):String
{
    return df.format(dateValue);
}
|
```

Chapitre 57 : Localisation d'applications

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La localisation consiste à inclure des actifs prenant en charge plusieurs jeux de paramètres régionaux. Les paramètres régionaux correspondent à une langue et un code de pays. Par exemple, fr_FR fait référence à l'anglais tel qu'il est parlé aux Etats-Unis et fr_FR, au français utilisé en France. Pour localiser une application en fonction de ces paramètres régionaux, vous proposeriez deux jeux d'actifs : un pour les paramètres fr_FR et l'autre pour les paramètres fr_FR.

Les paramètres régionaux peuvent partager des langues. Ainsi, fr_FR et en_GB (Grande Bretagne) sont des paramètres régionaux différents. Dans ce cas, les deux jeux de paramètres régionaux spécifient l'anglais, mais le code de pays indique qu'ils sont différents et n'utilisent donc pas nécessairement les mêmes actifs. Par exemple, une application utilisant les paramètres régionaux fr_FR comprendrait peut-être le mot « color », alors que ce mot serait épelé « colour » dans les paramètres régionaux en_GB. Par ailleurs, les devises correspondraient aux dollars ou aux livres Sterling, selon les paramètres régionaux, et les formats de date et d'heure seraient peut-être aussi différents.

Vous pouvez également fournir un jeu d'actifs pour une langue sans spécifier de code de pays. Ainsi, vous pouvez fournir des actifs en pour l'anglais et des actifs supplémentaires pour les paramètres régionaux fr_FR, qui sont spécifiques à l'anglais américain.

La localisation ne se limite pas à traduire les chaînes utilisées dans l'application. Elle peut également comprendre n'importe quel type d'actif, tels que les fichiers audio, les images ou les vidéos.

Choix d'un jeu de paramètres régionaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour déterminer les jeux de paramètres régionaux qu'utilise votre contenu ou votre application, vous pouvez procéder comme suit, au choix :

- Package flash.globalization : les classes gérant les paramètres régionaux du package flash.globalization permettent d'extraire les paramètres régionaux par défaut de l'utilisateur en fonction du système d'exploitation et des préférences de ce dernier. Cette solution est recommandée pour les applications qui tournent sur les moteurs d'exécution de Flash Player 10.1 ou ultérieur ou d'AIR 2.0 ou ultérieur. Pour plus d'informations, voir « Identification des paramètres régionaux » à la page 981.
- Invite utilisateur : vous pouvez démarrer l'application avec un jeu de paramètres régionaux par défaut et inviter l'utilisateur à choisir le jeu qu'il préfère.
- `(AIR uniquement) Capabilities.languages` : la propriété `Capabilities.languages` présente un tableau de langues disponible dans les langues préférées de l'utilisateur, telles qu'elles sont définies par le biais du système d'exploitation. Les chaînes contiennent des balises de langue (ainsi que des informations de script et de région, le cas échéant) définies par la norme RFC4646 (http://www.ietf.org/rfc/rfc4646.txt). Elles utilisent des tirets comme délimiteurs (par exemple, `"en-US"` ou `"ja-JP"`). La première entrée du tableau renvoyé possède le même identifiant de langue principale que la propriété language. Par exemple, si `languages[0]` est défini sur `"en-US"`, la propriété `language` est définie sur `"en"`. Cependant, si la propriété language est définie sur `"xu"` (qui représente une langue inconnue), le premier élément du tableau `languages` est différent.

- `Capabilities.language` : la propriété `Capabilities.language` indique le code de langue de l'interface utilisateur du système d'exploitation. Cette propriété est toutefois limitée à 20 langues connues. Sur les systèmes anglais, elle renvoie uniquement le code de langue et non le code du pays. C'est pourquoi il est préférable d'utiliser le premier élément du tableau `Capabilities.languages`.

Localisation de contenu Flex

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Adobe Flex propose une structure de localisation du contenu Flex. Cette structure contient les classes Locale, ResourceBundle et ResourceManagerImpl, ainsi que les interfaces IResourceBundle et IResourceManagerImpl.

Une bibliothèque de localisation Flex contenant des classes d'utilitaires destinés à trier les paramètres régionaux d'application est proposée sur Google Code (http://code.google.com/p/as3localelib/).

Voir aussi

http://code.google.com/p/as3localelib/

Localisation du contenu Flash

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Adobe Flash Professional comprend une classe Locale dans les composants ActionScript 3.0. La classe Locale vous permet de contrôler l'affichage de texte multilingue dans un fichier SWF. Le panneau Chaînes de Flash vous permet d'utiliser des ID de chaîne au lieu de littéraux de chaîne dans les champs de texte dynamique. Grâce à cette fonctionnalité, vous pouvez créer un fichier SWF qui affiche du texte chargé à partir d'un fichier XML spécifique à une langue. Pour plus d'informations sur l'utilisation de la classe Locale, voir Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Localisation d'applications AIR

Adobe AIR 1.0 et les versions ultérieures

Le SDK d'AIR propose une structure de localisation HTML (qui figure dans un fichier AIRLocalizer.js). Cette structure comprend des API qui facilitent la gestion de plusieurs paramètres régionaux dans une application HTML. Pour accéder à une bibliothèque ActionScript destinée à trier les paramètres régionaux, voir http://code.google.com/p/as3localelib/.

Voir aussi

http://code.google.com/p/as3localelib/

Localisation des dates, heures et devises

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'affichage des dates, heures et devises dans les applications varie grandement en fonction du jeu de paramètres régionaux. Aux Etats-Unis, par exemple, la date est représentée sous la forme mois/jour/année alors qu'en Europe, la norme consiste à utiliser jour/mois/année.

Vous pouvez écrire du code pour formater les dates, heures et devises. Par exemple, le code suivant convertit un objet Date du format mois/jour/année au format jour/mois/année. Si la variable `locale` (qui représente le jeu de paramètres régionaux) est définie sur `fr_FR`, la fonction renvoie le format mois/jour/année. L'exemple convertit un objet Date au format jour/mois/année pour tous les autres jeux de paramètres régionaux :

```
function convertDate(date)
{
    if (locale == "en_US")
    {
        return (date.getMonth() + 1) + "/" + date.getDate() + "/" + date.getFullYear();
    }
    else
    {
        return date.getDate() + "/" + (date.getMonth() + 1) + "/" + date.getFullYear();
    }
}
```

ADOBE FLEX

La structure Flex propose des contrôles de formatage de la date, des heures et des devises, notamment DateFormatter et CurrencyFormatter.

- mx:DateFormatter
- mx:CurrencyFormatter

Chapitre 58 : A propos de l'environnement HTML

Adobe AIR 1.0 et les versions ultérieures

Adobe® AIR® fait appel à WebKit *(www.webkit.org)*, également utilisé par le navigateur Web Safari, pour analyser, disposer et rendre un contenu HTML ou JavaScript. Il n'est pas indispensable d'utiliser les API AIR dans un contenu HTML. Vous avez la possibilité de programmer intégralement le contenu d'un objet HTMLLoader ou d'une fenêtre HTML à l'aide des langages HTML et JavaScript. La plupart des pages et applications HTML existantes devraient être exécutées en présentant peu de modifications (à condition qu'elles utilisent des fonctions HTML, CSS, DOM et JavaScript compatibles avec WebKit).

Important : les nouvelles versions du moteur d'exécution Adobe AIR comprennent parfois des versions mises à jour de WebKit. L'intégration d'une mise à jour de WebKit à une nouvelle version d'AIR *risque* d'engendrer des modifications inattendues dans une application AIR déployée. Ces modifications affectent parfois le comportement ou l'apparence du contenu HTML d'une application. Par exemple, les améliorations ou les corrections apportées au rendu WebKit ont parfois un impact sur les éléments de mise en forme de l'interface utilisateur d'une application. C'est pourquoi il est fortement recommandé d'intégrer un mécanisme de mise à jour à votre application. S'il s'avère nécessaire de mettre à jour votre application en raison d'une modification de la version de WebKit intégrée à AIR, le mécanisme de mise à jour d'AIR peut inviter l'utilisateur à installer la nouvelle version de votre application.

Le tableau suivant répertorie les versions du navigateur Web Safari qui font appel à la version de WebKit correspondant à celle utilisée dans AIR :

Version d'AIR	Version de Safari
1.0	2.04
1.1	3.04
1.5	4.0 bêta
2.0	4.03
2.5	4.03
2.6	4.03
2.7	4.03
3	5.0.3

Pour déterminer la version installée de WebKit, il suffit de vérifier la chaîne de l'agent utilisateur par défaut renvoyée par un objet HTMLLoader :

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

N'oubliez pas que la version de WebKit utilisée dans AIR n'est pas identique à la version open source. Certaines fonctions ne sont pas prises en charge dans AIR et la version d'AIR est susceptible d'inclure des correctifs liés à la sécurité et aux bogues qui ne sont pas encore intégrés à la version correspondante de WebKit. Voir « Fonctions de WebKit non prises en charge dans AIR » à la page 1015.

Etant donné que les applications AIR sont exécutées directement dans le poste de travail, en bénéficiant d'un accès complet au système de fichiers, le modèle de sécurité applicable au contenu HTML est plus strict que celui d'un navigateur Web standard. Dans AIR, seul le contenu chargé à partir du répertoire d'installation de l'application est placé dans le *sandbox de l'application*. Le sandbox de l'application dispose du niveau de privilège le plus élevé, l'autorisant à accéder aux API AIR. AIR place les autres contenus dans des sandbox distincts en fonction de leur origine. Les fichiers chargés à partir du système de fichiers sont placés dans un sandbox local tandis que ceux qui ont été chargés à partir du réseau à l'aide des protocoles http: ou https: sont dirigés vers un sandbox dépendant du domaine du serveur distant. L'accès à toute API AIR est interdit au contenu de ces sandbox qui ne sont pas des applications. Ce type de contenu est exécuté comme il le serait dans un navigateur Web classique.

Le contenu HTML visible dans AIR n'affiche pas de contenu SWF ou PDF si des paramètres alpha, de mise à l'échelle ou de transparence sont appliqués. Pour plus d'informations, voir « Eléments à prendre en compte lors du chargement d'un contenu SWF ou PDF dans une page HTML » à la page 1045 et « Transparence de la fenêtre » à la page 927.

Voir aussi

Webkit DOM Reference

Safari HTML Reference

Safari CSS Reference

www.webkit.org

Présentation de l'environnement HTML

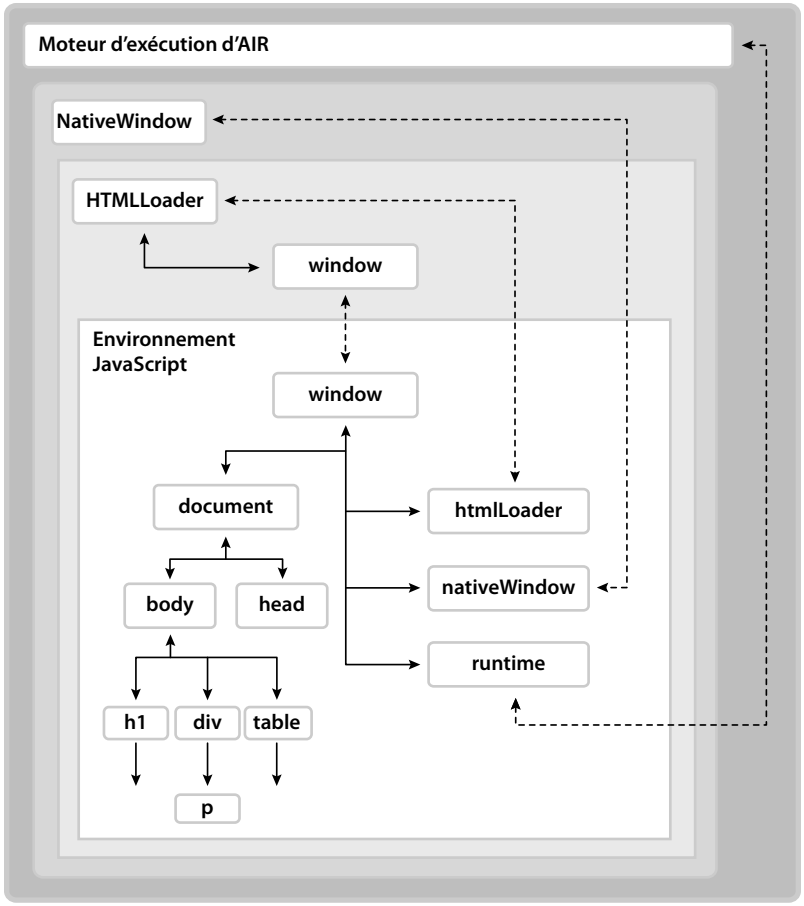
Adobe AIR 1.0 et les versions ultérieures

Adobe AIR propose un environnement JavaScript de type navigateur complet, doté d'une fonctionnalité de rendu HTML, d'un modèle d'objet de document et d'un interpréteur JavaScript. L'environnement JavaScript est représenté par la classe HTMLLoader d'AIR. Dans les fenêtres HTML, un objet HTMLLoader inclut tout le contenu HTML et est, à son tour, compris dans un objet NativeWindow. Dans un contenu SWF, il est possible d'ajouter la classe HTMLLoader, qui étend la classe Sprite, à la liste d'affichage d'une scène à l'instar de tout autre objet d'affichage. Les propriétés Adobe® ActionScript® 3.0 de la classe sont décrites à la section « Programmation du conteneur HTML d'AIR à l'aide de scripts » à la page 1043 et dans le manuel *Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash*. Dans la structure Flex, la classe AIR HTMLLoader est enveloppée dans un composant mx:HTML. Comme le composant mx:HTML étend la classe UIComponent, il peut être directement utilisé avec d'autres conteneurs Flex. L'environnement JavaScript au sein du composant mx:HTML ne présente aucune autre différence.

Présentation de l'environnement JavaScript et de sa relation avec l'hôte AIR

Adobe AIR 1.0 et les versions ultérieures

Le diagramme suivant illustre la relation existant entre l'environnement JavaScript et l'environnement du moteur d'exécution AIR. Bien qu'une seule fenêtre native soit affichée, une application AIR peut contenir plusieurs fenêtres. (De même, une seule fenêtre peut comprendre de multiples objets HTMLLoader.)



L'environnement JavaScript possède ses propres objets Document et Window. Le code JavaScript a des interactions avec l'environnement du moteur d'exécution AIR via les propriétés runtime, nativeWindow et htmlLoader. Quant au code ActionScript, il interagit avec l'environnement JavaScript par le biais de la propriété window d'un objet HTMLLoader, lequel fait référence à l'objet Window JavaScript. En outre, les objets ActionScript et JavaScript permettent d'écouter des événements envoyés à la fois par des objets AIR et JavaScript.

La propriété `runtime` offre un accès aux classes API AIR, ce qui vous permet de créer de nouveaux objets AIR de même que d'accéder aux membres de la classe (également appelée statique). Pour accéder à une API AIR, ajoutez le nom de la classe (package compris) à la propriété `runtime`. Par exemple, pour créer un objet File, utilisez l'instruction suivante :

```
var file = new window.runtime.filesystem.File();
```

Remarque : *le kit de développement d'AIR contient un fichier JavaScript, intitulé `AIRAliases.js`, qui définit des alias plus pratiques pour les classes AIR les plus courantes. Lors de l'importation de ce fichier, vous pouvez utiliser la forme abrégée air.Class au lieu de window.runtime.package.Class. Vous pourriez, par exemple, créer l'objet File à l'aide de `new air.File()`.*

L'objet NativeWindow offre des propriétés permettant de contrôler la fenêtre du poste de travail. A partir d'une page HTML, vous pouvez accéder à l'objet NativeWindow conteneur à l'aide de la propriété `window.nativeWindow`.

L'objet HTMLLoader propose des propriétés, des méthodes et des événements destinés à contrôler les modes de chargement et de rendu du contenu. A partir d'une page HTML, vous pouvez accéder à l'objet HTMLLoader parent à l'aide de la propriété `window.htmlLoader`.

Important : *seules les pages installées dans le cadre d'une application disposent des propriétés* `htmlLoader`, `nativeWindow` *ou* `runtime` *et ce, uniquement lorsqu'elles sont chargées en tant que document de niveau supérieur. Ces propriétés ne sont pas ajoutées dans le cas où le document est chargé dans une image ou une iframe. (Un document enfant peut accéder à ces propriétés dans le document parent du moment qu'il se trouve dans le même sandbox de sécurité. Par exemple, un document chargé dans un cadre pourrait accéder à la propriété* `runtime` *de son parent au moyen de* `parent.runtime`*.)*

A propos de la sécurité

Adobe AIR 1.0 et les versions ultérieures

AIR exécute la totalité du code au sein d'un sandbox de sécurité dépendant du domaine d'origine. Le contenu de l'application, qui se limite au contenu chargé à partir du répertoire d'installation de l'application, est placé dans le sandbox de l'*application*. L'accès à l'environnement d'exécution et aux API AIR est uniquement disponible pour les scripts HTML et JavaScript exécutés dans ce sandbox. Parallèlement à cela, les opérations d'évaluation et d'exécution dynamiques de code JavaScript sont en grande partie bloquées dans le sandbox de l'application une fois que tous les gestionnaires de l'événement `load` de la page ont été renvoyés.

Vous avez la possibilité de mapper une page d'application dans un sandbox autre que d'application. Pour ce faire, chargez la page dans une image ou une iframe et définissez les attributs `sandboxRoot` et`documentRoot` du cadre propres à AIR. En configurant la valeur de `sandboxRoot` sur un domaine distant réel, vous permettez au contenu placé dans le sandbox d'intercoder le contenu de ce domaine. Cette méthode de mappage de pages peut s'avérer pratique lors du chargement et du codage du contenu distant, dans le cadre d'une application *composite* par exemple.

Une autre solution permettant d'intercoder des contenus d'application et autre (et la seule manière de donner accès à des API AIR à un contenu autre que d'application) consiste à créer un *pont de sandbox*. Un pont *parent-enfant* permet au contenu d'une image enfant, d'une iframe ou d'une fenêtre d'accéder à des méthodes et propriétés désignées définies dans le sandbox de l'application. A l'inverse, un pont *enfant-parent* permet au contenu de l'application d'accéder à des méthodes et propriétés désignées définies dans le sandbox de l'enfant. Pour définir un pont de sandbox, vous devez définir les propriétés `parentSandboxBridge` et `childSandboxBridge` de l'objet window. Pour plus d'informations, voir « Sécurité HTML dans Adobe AIR » à la page 1127 et « Eléments image et iframe HTML » à la page 1011.

A propos des modules d'extension et des objets incorporés

Adobe AIR 1.0 et les versions ultérieures

AIR prend en charge le module d'extension Adobe® Acrobat®. Pour afficher le contenu PDF, les utilisateurs doivent disposer d'Acrobat ou d'Adobe® Reader® 8.1 (ou version ultérieure). L'objet HTMLLoader comprend une propriété permettant de vérifier si le système d'un utilisateur peut afficher ou non des documents PDF. Il est également possible d'afficher le contenu des fichiers SWF au sein de l'environnement HTML. Cette fonction étant intégrée dans AIR, elle ne fait appel à aucun module d'extension.

AIR ne prend en charge aucun autre module d'extension Webkit.

Voir aussi

« Sécurité HTML dans Adobe AIR » à la page 1127

« Sandbox HTML » à la page 1003

« Eléments image et iframe HTML » à la page 1011

« Objet Window JavaScript » à la page 1009

« Objet XMLHttpRequest » à la page 1005

« Ajout d'un contenu PDF dans AIR » à la page 569

AIR et WebKit

Adobe AIR 1.0 et les versions ultérieures

Adobe AIR a recours au moteur Webkit, disponible en open source et également utilisé dans le navigateur Web Safari. AIR ajoute plusieurs extensions pour accéder aux objets et aux classes d'exécution ainsi que pour des raisons de sécurité. En outre, Webkit lui-même insère des fonctions non incluses dans les normes W3C des langages HTML, CSS et JavaScript.

Seuls les ajouts d'AIR et les extensions Webkit les plus significatives sont traités dans cette section. Pour obtenir des informations complémentaires sur les langages HTML, CSS et JavaScript non standard, voir www.webkit.org et developer.apple.com. Pour plus d'informations sur les normes, consulter le site Web W3C. Mozilla propose également une page de référence très intéressante portant sur les technologies HTML, CSS et DOM (bien évidemment, les moteurs Webkit et Mozilla sont différents l'un de l'autre).

Code JavaScript dans AIR

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

AIR apporte plusieurs modifications au comportement classique des objets JavaScript courants. La plupart de ces modifications sont destinées à simplifier l'écriture d'applications sécurisées dans AIR. Parallèlement à cela, ces différences de comportement impliquent que certains modèles de codage JavaScript courants et les applications Web existantes qui s'en servent risquent de ne pas fonctionner comme prévu dans AIR. Pour plus d'informations sur la correction de ces types de problèmes, voir la section « Contournement des erreurs JavaScript liées à la sécurité » à la page 1021.

Sandbox HTML

Adobe AIR 1.0 et les versions ultérieures

AIR place les différents contenus dans des sandbox distincts en fonction de leur origine. Les règles de sandbox respectent la stratégie de « même origine » implémentée par la plupart des navigateurs Web, de même que les règles applicables aux sandbox mises en œuvre par Adobe Flash Player. AIR propose en outre un nouveau type de sandbox d'*application* conçu pour contenir et protéger le contenu de l'application. Pour plus d'informations sur les types de sandbox que vous êtes susceptible de rencontrer lors du développement d'applications AIR, voir « Sandbox de sécurité » à la page 1087.

L'accès à l'environnement d'exécution et aux API AIR est uniquement disponible pour les scripts HTML et JavaScript exécutés dans le sandbox de l'application. Parallèlement à cela, toutefois, les opérations d'évaluation et d'exécution dynamiques de code JavaScript, sous leurs diverses formes, sont largement limitées au sein du sandbox de l'application pour des raisons de sécurité. Ces restrictions s'appliquent que votre application charge réellement ou non des informations directement depuis un serveur. (Même le contenu des fichiers, les chaînes collées et les données saisies directement par l'utilisateur peuvent ne pas être fiables.)

L'origine du contenu d'une page détermine le sandbox associé au contenu. Seul le contenu chargé à partir du répertoire de l'application (le répertoire d'installation auquel le modèle d'URL `app:` fait référence) est placé dans le sandbox de l'application. Tout contenu chargé à partir du système de fichiers est placé dans le sandbox *local avec système de fichiers* ou le sandbox de sécurité *approuvé localement*, lequel permet d'accéder au contenu situé sur le système de fichiers local (mais pas au contenu distant) et d'interagir avec lui. Tout contenu chargé à partir du réseau est dirigé vers un sandbox distant correspondant à son domaine d'origine.

Pour qu'une page d'application puisse interagir librement avec un contenu situé dans un sandbox distant, vous pouvez la mapper au même domaine que le contenu distant. Si, par exemple, vous écrivez une application affichant des données de mappage provenant d'un service Internet, la page de l'application qui est chargée et qui présente le contenu émanant du service pourrait être mappée au domaine du service. Les attributs de mappage des pages dans un sandbox et un domaine distants sont de nouveaux attributs ajoutés aux éléments image et iframe HTML.

Pour permettre à un contenu d'un sandbox autre que d'application d'utiliser les fonctions d'AIR en toute sécurité, vous pouvez configurer un pont de sandbox parent. Pour permettre au contenu de l'application d'appeler des méthodes et des propriétés d'accès de contenu en toute sécurité dans d'autres sandbox, configurez un pont de sandbox enfant. La notion de sécurité évoquée ici signifie que le contenu distant ne peut pas obtenir accidentellement de références à des objets, des propriétés ou des méthodes exposés de manière non explicite. Seuls les objets anonymes, fonctions et types de données simples peuvent être transmis via le pont. Vous devez néanmoins éviter d'exposer explicitement des fonctions potentiellement dangereuses. Si, par exemple, vous avez exposé une interface qui autorisait le contenu distant à lire et à écrire des fichiers n'importe où sur le système d'un utilisateur, vous risquez de donner à ce contenu les moyens de nuire considérablement aux utilisateurs.

Fonction eval() JavaScript

Adobe AIR 1.0 et les versions ultérieures

L'utilisation de la fonction `eval()` est limitée au contenu du sandbox de l'application une fois le téléchargement de la page terminé. Dans certains cas, cette fonction peut servir à analyser en toute sécurité des données au format JSON, mais toute évaluation aboutissant à des résultats d'instructions exécutables se traduit par une erreur. La section « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130 décrit les utilisations autorisées pour la fonction `eval()`.

Constructeurs de fonctions

Adobe AIR 1.0 et les versions ultérieures

Dans le sandbox d'application, il est possible d'utiliser des constructeurs de fonctions avant la fin du chargement d'une page. Dès lors que tous les gestionnaires d'événement `load` de page sont terminés, il est impossible de créer de nouvelles fonctions.

Chargement d'un script externe

Adobe AIR 1.0 et les versions ultérieures

Les pages HTML du sandbox de l'application ne peuvent pas utiliser la balise `script` pour charger des fichiers JavaScript situés en dehors du répertoire de l'application. Pour qu'une page de votre application puisse charger un script stocké en dehors du répertoire de l'application, vous devez la mapper dans un sandbox autre que celui de d'application.

Objet XMLHttpRequest

Adobe AIR 1.0 et les versions ultérieures

AIR fournit un objet XMLHttpRequest (XHR) dont les applications peuvent se servir pour émettre des requêtes de données. L'exemple suivant illustre une demande de données simple :

```
xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", "http:/www.example.com/file.data", true);
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4) {
        //do something with data...
    }
}
xmlhttp.send(null);
```

Contrairement à un navigateur, AIR permet au contenu exécuté dans le sandbox de l'application de demander des données provenant de n'importe quel domaine. Le résultat d'une requête XHR contenant une chaîne JSON peut aboutir à des objets de données à moins de comprendre également du code exécutable. Si des instructions exécutables figurent dans le résultat de la requête XHR, une erreur est renvoyée et la tentative d'évaluation se solde par un échec.

Pour empêcher l'introduction accidentelle de code provenant de sources distantes, les requêtes XHR synchrones renvoient un résultat vide lorsqu'elles sont émises avant la fin du chargement de la page. Les requêtes XHR asynchrones sont toujours renvoyées après le chargement d'une page.

Par défaut, AIR bloque les requêtes XMLHttpRequest interdomaines dans les sandbox autres que d'application. Une fenêtre parent du sandbox de l'application peut choisir d'autoriser des requêtes interdomaines dans une image enfant dont le contenu ne se trouve pas dans un sandbox autre que d'application. Pour ce faire, définissez `allowCrossDomainXHR`, un attribut ajouté par AIR, sur la valeur `true` dans l'élément image ou iframe conteneur :

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

Remarque : *si cela s'avère pratique, il est aussi possible de télécharger les données à l'aide de la classe URLStream AIR.*

Si vous distribuez une requête XMLHttpRequest vers un serveur distant à partir d'une image ou d'une iframe doté du contenu de l'application mappé dans un sandbox distant, assurez-vous que l'URL de mappage ne masque pas l'adresse du serveur utilisé dans la requête XHR. Tenez compte, par exemple, de la définition d'iframe suivante, laquelle mappe le contenu de l'application dans un sandbox distant pour le domaine example.com :

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

Etant donné que l'attribut `sandboxRoot` remappe l'URL racine de l'adresse www.example.com, toutes les requêtes sont chargées à partir du répertoire de l'application et non du serveur distant. Les requêtes sont remappées, qu'elles soient issues de la navigation dans les pages ou d'une requête XMLHttpRequest.

Afin d'éviter de bloquer accidentellement des requêtes de données en direction du serveur distant, mappez `sandboxRoot` à un sous-répertoire de l'URL distante plutôt qu'à la racine. Le répertoire ne doit pas nécessairement exister. Par exemple, pour permettre le chargement des requêtes en direction de www.example.com à partir du serveur distant plutôt que du répertoire de l'application, modifiez l'iframe précédente de la manière suivante :

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

Dans ce cas, seul le contenu du sous-répertoire `air` est chargé en local.

Pour plus d'informations sur le mappage de sandbox, voir « Eléments image et iframe HTML » à la page 1011 et « Sécurité HTML dans Adobe AIR » à la page 1127.

Cookies

Adobe AIR 1.0 et les versions ultérieures

Dans les applications AIR, seul le contenu des sandbox distants (chargé à partir de sources http: et https:) peut utiliser des cookies (propriété `document.cookie`). Le sandbox d'application propose d'autres techniques de stockage des données persistantes, telles que les classes EncryptedLocalStore, SharedObject et FileStream.

Objet Clipboard

Adobe AIR 1.0 et les versions ultérieures

L'API Clipboard de WebKit est dotée des événements suivants : `copy`, `cut` et `paste`. L'objet événement transmis dans ces événements permet d'accéder au Presse-papiers via la propriété `clipboardData`. Faites appel aux méthodes suivantes de l'objet `clipboardData` pour lire ou écrire des données de Presse-papiers :

Méthode	Description
clearData(mimeType)	Efface les données du Presse-papiers. Définissez le paramètre `mimeType` sur le type MIME des données à effacer.
getData(mimeType)	Permet d'obtenir les données du Presse-papiers. Cette méthode peut uniquement être appelée dans un gestionnaire pour l'événement `paste`. Définissez le paramètre `mimeType` sur le type MIME des données à renvoyer.
setData(mimeType, data)	Copie les données dans le Presse-papiers. Définissez le paramètre `mimeType` sur le type MIME des données.

Le code JavaScript situé en dehors du sandbox de l'application peut uniquement accéder au Presse-papiers par le biais de ces événements. Toutefois, le contenu du sandbox de l'application a la possibilité d'accéder directement au Presse-papiers du système à l'aide de la classe Clipboard AIR. Par exemple, l'instruction suivante pourrait servir à obtenir des données au format texte du Presse-papiers :

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                    air.ClipboardTransferMode.ORIGINAL_ONLY);
```

Les types MIME de données valides sont les suivants :

Type MIME	Valeur
Texte	"text/plain"
HTML	"text/html"

Type MIME	Valeur
URL	"text/uri-list"
Image bitmap	"image/x-vnd.adobe.air.bitmap"
Liste de fichiers	"application/x-vnd.adobe.air.file-list"

Important : *seul le contenu du sandbox de l'application peut accéder aux données de fichier stockées dans le Presse-papiers. Si du contenu hors application tente d'accéder à un objet fichier du Presse-papiers, une erreur de sécurité est générée.*

Pour plus d'informations sur l'utilisation du Presse-papiers, voir « Opération de copier-coller » à la page 616 et Using the Pasteboard from JavaScript (centre des développeurs Apple).

Opération glisser-déposer

Adobe AIR 1.0 et les versions ultérieures

Les mouvements de glisser-déposer vers et depuis un contenu HTML génèrent les événements DOM suivants : `dragstart`, `drag`, `dragend`, `dragenter`, `dragover`, `dragleave` et `drop`. L'objet événement transmis dans ces événements permet d'accéder aux données déplacées via la propriété `dataTransfer`. La propriété `dataTransfer` fait référence à un objet offrant les mêmes méthodes que l'objet `clipboardData` associé à un événement clipboard. Par exemple, la fonction suivante pourrait servir à obtenir des données au format texte à partir d'un événement `drop` :

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

L'objet `dataTransfer` dispose des membres importants suivants :

Membre	Description
clearData(mimeType)	Efface les données. Définissez le paramètre `mimeType` sur le type MIME de la représentation de données à effacer.
getData(mimeType)	Permet d'obtenir les données que vous avez fait glisser. Cette méthode peut uniquement être appelée dans un gestionnaire pour l'événement `drop`. Définissez le paramètre `mimeType` sur le type MIME des données à obtenir.
setData(mimeType, data)	Définit les données à faire glisser. Définissez le paramètre `mimeType` sur le type MIME des données.
types	Tableau de chaînes contenant les types MIME de toutes les représentations de données actuellement disponibles dans l'objet `dataTransfer`.
effectsAllowed	Indique si les données que vous faites glisser peuvent être copiées, déplacées et/ou liées. Définissez la propriété `effectsAllowed` du gestionnaire pour l'événement `dragstart`.
dropEffect	Indique, parmi les effets de type drop (déposer) autorisés, ceux qui sont pris en charge par une cible de type drag (glisser). Définissez la propriété `dropEffect` du gestionnaire pour l'événement `dragEnter`. Au cours du glissement, le curseur change de forme afin d'indiquer l'effet qui se produirait si l'utilisateur relâchait le bouton de la souris. Si aucun effet `dropEffect` n'est spécifié, un effet doté de la propriété `effectsAllowed` est appliqué. L'effet copy (copier) est prioritaire sur l'effet move (déplacer), lequel a priorité sur l'effet link (lier). L'utilisateur peut modifier le niveau de priorité par défaut au moyen du clavier.

Pour plus d'informations sur l'ajout de la prise en charge du glisser-déposer à une application AIR, voir « Opération glisser-déposer dans AIR » à la page 629 et Using the Drag-and-Drop from JavaScript (centre de développeurs Apple).

Propriétés innerHTML et outerHTML

Adobe AIR 1.0 et les versions ultérieures

Pour des raisons de sécurité, AIR limite l'utilisation des propriétés `innerHTML` et `outerHTML` avec un contenu exécuté dans le sandbox de l'application. Avant l'événement de chargement de la page, de même que pendant l'exécution de tout gestionnaire d'événement de chargement, l'utilisation des propriétés `innerHTML` et `outerHTML` n'est pas restreinte. Toutefois, dès lors que la page est chargée, vous pouvez uniquement vous servir des propriétés `innerHTML` ou `outerHTML` afin d'ajouter un contenu statique au document. Toute instruction de la chaîne attribuée à `innerHTML` ou à `outerHTML` qui aboutit à du code exécutable est ignorée. Si, par exemple, vous incluez un attribut de rappel d'événement dans une définition d'élément, l'écouteur d'événement n'est pas inséré. De la même manière, les balises `<script>` intégrées ne sont pas évaluées. Pour plus d'informations, voir « Sécurité HTML dans Adobe AIR » à la page 1127.

Méthodes Document.write() et Document.writeln()

Adobe AIR 1.0 et les versions ultérieures

L'utilisation des méthodes `write()` et `writeln()` n'est pas limitée dans le sandbox de l'application avant la survenue de l'événement `load` de la page. Toutefois, dès lors que la page est chargée, l'appel de l'une ou l'autre de ces méthodes n'efface pas la page ou n'en crée pas de nouvelle. Dans un sandbox autre que celui de l'application, comme dans la plupart des navigateurs Web, l'appel de la méthode `document.write()` ou `writeln()` après la fin du chargement de la page entraîne l'effacement de la page active et l'ouverture d'une nouvelle page vide.

Propriété Document.designMode

Adobe AIR 1.0 et les versions ultérieures

Définissez la propriété `document.designMode` sur la valeur `on` afin de rendre tous les éléments du document modifiables. L'éditeur intégré prend en charge les modifications de texte, la copie, le collage et le glisser-déposer. Définir `designMode` sur `on` revient au même que configurer la propriété `contentEditable` de l'élément `body` sur la valeur `true`. La propriété `contentEditable` permet de définir les sections modifiables d'un document pour la plupart des éléments HTML. Pour plus d'informations, voir la section « Attribut contentEditable HTML » à la page 1014.

Evénements unload (pour les objets body et frameset)

Adobe AIR 1.0 et les versions ultérieures

Dans la balise `frameset` ou `body` de niveau supérieur d'une fenêtre (y compris la fenêtre principale de l'application), ne répondez pas à la fenêtre (ou à l'application) en cours de fermeture au moyen de l'événement `unload`. Au lieu de cela, optez pour l'événement `exiting` de l'objet NativeApplication (afin de détecter le moment de fermeture d'une application). Une autre solution consiste à utiliser l'événement `closing` de l'objet NativeWindow (afin de détecter le moment de fermeture d'une fenêtre). Par exemple, le code JavaScript suivant affichage un message (`« Goodbye »`) lorsque l'utilisateur ferme l'application :

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

Toutefois, les scripts *peuvent* réagir correctement à l'événement `unload` déclenché par la navigation dans une image, une iframe ou le contenu d'une fenêtre de niveau supérieur.

Remarque : *il est probable que ces limitations soient supprimées dans une prochaine version d'Adobe AIR.*

Objet Window JavaScript

Adobe AIR 1.0 et les versions ultérieures

L'objet Window reste l'objet global dans le contexte d'exécution JavaScript. Dans le sandbox de l'application, AIR insère de nouvelles propriétés dans l'objet Window JavaScript en vue d'offrir un accès aux classes intégrées d'AIR de même qu'à des objets hôte importants. En outre, certaines méthodes et propriétés se comportent différemment selon qu'elles se trouvent dans le sandbox de l'application ou pas.

Propriété Window.runtime La propriété `runtime` vous permet d'instancier et d'utiliser les classes d'exécution intégrées à partir du sandbox de l'application. Ces classes comprennent les API AIR et Flash Player (mais pas, par exemple, la structure Flex). L'instruction suivante crée par exemple un objet de fichier AIR :

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

Le fichier `AIRAliases.js`, disponible dans le kit SDK d'AIR, contient des définitions d'alias permettant de raccourcir de telles références. Ainsi, lorsque le fichier `AIRAliases.js` est importé dans une page, il est possible de créer un objet File à l'aide de l'instruction suivante :

```
var preferencesFile = new air.File();
```

La propriété `window.runtime` est exclusivement définie pour un contenu situé dans le sandbox de l'application et uniquement pour le document parent d'une page dotée d'images ou d'iframes.

Voir la section « Utilisation du fichier AIRAliases.js » à la page 1027.

Propriété Window.nativeWindow La propriété `nativeWindow` fournit une référence à l'objet window natif sous-jacent. Grâce à cette propriété, vous pouvez coder des fonctions et propriétés de fenêtre (window) telles que la position, la taille et la visibilité, et gérer des événements de fenêtre comme la fermeture, le redimensionnement et le déplacement. Dans l'exemple suivant, l'instruction permet de fermer la fenêtre :

```
window.nativeWindow.close();
```

Remarque : *les fonctions de contrôle de la fenêtre fournies par l'objet NativeWindow chevauchent celles fournies par l'objet Window JavaScript. Dans de tels cas, vous pouvez opter pour la méthode qui vous semble la plus pratique.*

La propriété `window.nativeWindow` est exclusivement définie pour un contenu situé dans le sandbox de l'application et pour le document parent d'une page dotée d'images ou d'iframes.

Propriété Window.htmlLoader La propriété `htmlLoader` fournit une référence à l'objet HTMLLoader AIR disposant du contenu HTML. Grâce à cette propriété, vous pouvez coder l'aspect et le comportement de l'environnement HTML. Vous pouvez ainsi utiliser la propriété `htmlLoader.paintsDefaultBackground` afin de déterminer si la commande dessine un arrière-plan blanc par défaut :

```
window.htmlLoader.paintsDefaultBackground = false;
```

Remarque : *l'objet HTMLLoader proprement dit est doté d'une propriété* `window`*, laquelle fait référence à l'objet Window JavaScript du contenu HTML inclus. Cette propriété vous permet d'accéder à l'environnement JavaScript via une référence à l'objet HTMLLoader conteneur.*

La propriété `window.htmlLoader` est exclusivement définie pour un contenu situé dans le sandbox de l'application et pour le document parent d'une page dotée d'images ou d'iframes.

Propriétés Window.parentSandboxBridge et Window.childSandboxBridge Les propriétés `parentSandboxBridge` et `childSandboxBridge` vous permettent de définir une interface entre un cadre parent et un cadre enfant. Pour plus d'informations, voir la section « Programmation croisée du contenu dans des sandbox de sécurité distincts » à la page 1038.

Fonctions Window.setTimeout() et Window.setInterval() Pour des raisons de sécurité, AIR limite l'utilisation des fonctions `setTimeout()` et `setInterval()` dans le sandbox de l'application. Il est impossible de définir le code à exécuter sous forme de chaîne lors de l'appel de la fonction `setTimeout()` ou `setInterval()`. Vous devez impérativement utiliser une référence à la fonction. Pour plus d'informations, voir la section « setTimeout() et setInterval() » à la page 1024.

Fonction Window.open() Lorsqu'elle est appelée par du code exécuté dans un sandbox autre que d'application, la méthode `open()` ouvre seulement une fenêtre si elle est appelée suite à une action utilisateur (telle qu'un clic de souris ou l'activation d'une touche du clavier). De plus, le titre de la fenêtre est précédé du titre de l'application (afin d'empêcher l'ouverture de fenêtres par du contenu distant provenant de fenêtres d'usurpation d'identité ouvertes par l'application). Pour plus d'informations, voir la section « Restrictions relatives à l'appel de la méthode window.open() de JavaScript » à la page 1133.

Objet air.NativeApplication

Adobe AIR 1.0 et les versions ultérieures

L'objet NativeApplication fournit des informations sur l'état de l'application, distribue différents événements de niveau application importants et offre des fonctions pratiques de contrôle du comportement de l'application. Une occurrence unique de l'objet NativeApplication est créée automatiquement. Elle est accessible via la propriété `NativeApplication.nativeApplication` définie dans la classe.

Pour accéder à l'objet à partir du code JavaScript, vous pouvez utiliser :

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

Autre solution, si le script `AIRAliases.js` a été importé, vous adoptez la forme plus courte suivante :

```
var app = air.NativeApplication.nativeApplication;
```

L'objet NativeApplication est uniquement accessible à partir du sandbox de l'application. Pour plus d'informations sur l'objet NativeApplication, voir « Utilisation des informations sur le moteur d'exécution d'AIR et les systèmes d'exploitation » à la page 920.

Modèle d'URL JavaScript

Adobe AIR 1.0 et les versions ultérieures

L'exécution d'un code défini dans un modèle d'URL JavaScript (comme dans `href="javascript:alert('Test')"`) est bloquée au sein du sandbox de l'application. Aucune erreur n'est renvoyée.

HTML dans AIR

Adobe AIR 1.0 et les versions ultérieures

AIR et WebKit définissent deux éléments et attributs HTML non standard, à savoir :

« Eléments image et iframe HTML » à la page 1011

« Gestionnaires d'événement d'élément HTML » à la page 1013

Eléments image et iframe HTML

Adobe AIR 1.0 et les versions ultérieures

AIR ajoute de nouveaux attributs aux éléments image et iframe de contenu dans le sandbox de l'application :

Attribut sandboxRoot L'attribut `sandboxRoot` spécifie un domaine de remplacement autre que d'application d'origine pour le fichier indiqué par l'attribut `src` du cadre. Le fichier est chargé dans le sandbox autre que d'application correspondant au domaine précisé. Le contenu du fichier et celui chargé à partir du domaine indiqué peuvent être intercodés.

Important : *si vous définissez la valeur de `sandboxRoot` sur l'URL de base du domaine, toutes les requêtes de contenu provenant de ce domaine sont chargées depuis le répertoire de l'application au lieu du serveur distant (qu'elles résultent d'une opération de navigation dans les pages, d'une requête XMLHttpRequest ou d'une autre méthode de chargement de contenu).*

Attribut documentRoot L'attribut `documentRoot` indique le répertoire local à partir duquel sont chargées les URL renvoyant à des fichiers situés à l'emplacement spécifié par `sandboxRoot`.

Lors de la résolution des URL, soit dans l'attribut `src` du cadre soit dans le contenu chargé dans le cadre, la partie de l'URL correspondant à la valeur précisée dans `sandboxRoot` est remplacée par celle définie dans `documentRoot`. Par conséquent, dans la balise de cadre suivante :

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html` est chargé depuis le sous-répertoire `sandbox` du dossier d'installation de l'application. Les URL relatives contenues dans `child.html` sont résolues en fonction du répertoire `sandbox`. Vous noterez que les fichiers situés sur le serveur distant à l'adresse `www.example.com/air` ne sont pas accessibles dans le cadre, car AIR tenterait sinon de les charger à partir du répertoire app:/sandbox/.

Attribut allowCrossDomainXHR Incluez `allowCrossDomainXHR="allowCrossDomainXHR"` dans la balise de cadre d'ouverture afin de permettre au contenu du cadre d'émettre des requêtes XMLHttpRequest en direction de tout domaine distant. Par défaut, un contenu autre que d'application peut uniquement effectuer de telles requêtes en direction de son propre domaine d'origine. La possibilité d'effectuer des XHR interdomaines entraîne des risques importants au niveau de la sécurité. Le code de la page permet en effet d'échanger des données avec n'importe quel domaine. Si un contenu malveillant est introduit dans la page d'une manière quelconque, toutes les données accessibles au code dans le sandbox actif sont en danger. C'est pourquoi il est recommandé d'activer uniquement les requêtes XHR interdomaines pour les pages que vous créez et contrôlez, et à condition que le chargement de données interdomaines soit réellement nécessaire. En outre, validez avec précaution toutes les données externes chargées par la page afin d'empêcher l'introduction de code ou d'autres formes d'attaques.

Important : *si l'attribut `allowCrossDomainXHR` se trouve dans un élément image ou iframe, les requêtes XHR interdomaines sont activées (à moins que la valeur attribuée soit égale à 0 ou commence par les lettres f ou n). Par exemple, définir `allowCrossDomainXHR` sur la valeur `deny` maintient l'activation des requêtes XHR interdomaines. Excluez totalement l'attribut de la déclaration d'élément si vous préférez ne pas activer les requêtes interdomaines.*

ondominitialize, attribut Indique un gestionnaire d'événement pour l'événement `dominitialize` d'un cadre. Cet événement, spécifique à AIR, est déclenché une fois que les objets window et document du cadre ont été créés, mais avant l'analyse des scripts ou la création d'éléments de document.

Le cadre distribue l'événement `dominitialize` suffisamment tôt dans la séquence de chargement pour qu'un script de la page enfant puisse référencer les objets, variables et fonctions ajoutés au document enfant par le gestionnaire `dominitialize`. La page parent doit se trouver dans le même sandbox que l'enfant pour pouvoir insérer ou ouvrir

directement tout objet figurant dans un document enfant. Toutefois, un parent situé dans le sandbox de l'application peut établir un pont de sandbox afin de communiquer avec le contenu d'un sandbox autre que d'application.

Les exemples suivants illustrent l'utilisation de la balise iframe dans AIR :

Placez `child.html` dans un sandbox distant, sans définir de mappage à un domaine réel situé sur un serveur distant :

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

Placez `child.html` dans un sandbox distant, en autorisant uniquement les requêtes XMLHttpRequest émises en direction de `www.example.com` :

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

Placez `child.html` dans un sandbox distant, en autorisant les requêtes XMLHttpRequest émises en direction de n'importe quel domaine distant :

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

Placez `child.html` dans un sandbox local avec système de fichiers :

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

Placez `child.html` dans un sandbox distant en vous servant de l'événement `dominitialize` pour établir un pont de sandbox :

```
<html>
<head>
<script>
var bridgeInterface = {};
bridgeInterface.testProperty = "Bridge engaged";
function engageBridge(){
    document.getElementById("sandbox").parentSandboxBridge = bridgeInterface;
}
</script>
</head>
<body>
<iframe id="sandbox"
            src="http://www.example.com/air/child.html"
            documentRoot="app:/"
            sandboxRoot="http://www.example.com/air/"
            ondominitialize="engageBridge()"/>
</body>
</html>
```

Le document `child.html` suivant montre comment un contenu enfant peut accéder au pont de sandbox parent :

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Pour plus d'informations, voir « Programmation croisée du contenu dans des sandbox de sécurité distincts » à la page 1038 et « Sécurité HTML dans Adobe AIR » à la page 1127.

Gestionnaires d'événement d'élément HTML

Adobe AIR 1.0 et les versions ultérieures

Dans AIR et Webkit, les objets DOM distribuent certains événements introuvables dans le modèle d'événement DOM standard. Le tableau suivant dresse la liste des attributs d'événement associés que vous pouvez utiliser afin de spécifier des gestionnaires pour ces événements :

Nom de l'attribut de rappel	Description
oncontextmenu	Appelé lorsqu'un menu contextuel est invoqué, comme par le biais d'un clic droit ou d'un clic associé à une commande sur le texte sélectionné.
oncopy	Appelé lorsqu'une sélection d'un élément est copiée.
oncut	Appelé lorsqu'une sélection d'un élément est coupée.
ondominitialize	Appelé lorsque le DOM d'un document chargé dans une image ou une iframe est créé, mais avant que tout élément DOM ne soit créé ou que les scripts soient analysés.
ondrag	Appelé lorsqu'un élément est déplacé par glissement.
ondragend	Appelé lorsqu'un glissement est relâché.
ondragenter	Appelé lorsqu'un mouvement de glissement entre dans les limites d'un élément.
ondragleave	Appelé lorsqu'un mouvement de glissement quitte les limites d'un élément.
ondragover	Appelé continuellement lorsqu'un mouvement de glissement se trouve dans les limites d'un élément.
ondragstart	Appelé lorsqu'un mouvement de glissement commence.
ondrop	Appelé lorsqu'un mouvement de glissement est relâché au-dessus d'un élément.
onerror	Appelé lorsqu'une erreur se produit au cours du chargement d'un élément.
oninput	Appelé lorsque du texte est saisi dans un élément de formulaire.
onpaste	Appelé lorsqu'un élément est collé dans un élément.
onscroll	Appelé lorsque le contenu d'un élément qui défile est en train de défiler.
onselectstart	Appelé lorsqu'une sélection commence.

Attribut contentEditable HTML

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez ajouter l'attribut `contentEditable` à tout élément HTML permettant aux utilisateurs de modifier le contenu de l'élément. Par exemple, dans l'exemple suivant, le code HTML définit la totalité du document comme étant modifiable, à l'exception du premier élément `p` :

```
<html>
<head/>
<body contentEditable="true">
    <h1>de Finibus Bonorum et Malorum</h1>
    <p contentEditable="false">Sed ut perspiciatis unde omnis iste natus error.</p>
    <p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis.</p>
</body>
</html>
```

Remarque : *si vous définissez la propriété `document.designMode` sur la valeur `on`, tous les éléments du document deviennent modifiables, quelle que soit la configuration de l'attribut `contentEditable` d'un élément particulier. Toutefois, la définition de `designMode` sur la valeur `off` ne désactive pas la modification des éléments pour lesquels l'attribut `contentEditable` est défini sur `true`. Pour plus d'informations, voir la section «* *Propriété Document.designMode » à la page 1008.*

URL de type data:

Adobe AIR 2 et ultérieur

AIR prend en charge les URL de type `data:` pour les éléments suivants :

- img
- input type=”image”
- Règles CSS qui autorisent les images (telles les images d'arrière-plan)

Les URL de type data: permettent d'insérer directement des données d'images binaires dans un document CSS ou HTML sous forme de chaîne codée au format base64. L'exemple suivant utilise une URL de type data: sous forme d'arrière-plan répété :

```
<html>
<head>
<style>
body {
background-
image:url('
0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAAZQTFRF%2F6cA%2F%2F%2F%2Fgxp3lwAAAAJ0Uk5T%2FwDltzBKAAA
BF0lEQVR42uzZQQ7CMAxE0e%2F7X5oNCyRocWzPiJbMBZ6qpIljE%2BnwklgKG7kwUjc2IkIaxkY0CPdEsCCasws6ShX
BgmBBmEagpXQQLAgWBAuSY2gaKaWPYEGwIEwg0FRmECwIFoQeQjJlhJWUEFazjFDJCkI5WYRWMgjtfEGYyQnCXD4jTCd
m1zmngFpBFznwVNi5RPSbwbWnpYr%2BBHi%2FtCTfgPLEPL7jBctAKBRptXJ8M%2BprIuZKu%2BUKcg4YK1PLz7kx4bS
qHyPaT4d%2B28OCJJiRBo4FCQsSA0bziT3XubMgYUG6fc5fatmGBQkL0hoJ1IaZMiQsSFiQ8vRscTjlQOI2iHZwtpHuf
%2BJAYiOiJSkj8Z%2FIQ4ABANvXGLd3%2BZMrAAAAAElFTkSuQmCC');
background-repeat:repeat;
}
</style>
</head>
<body>
</body>
</html>
```

Lorsque vous utilisez des URL de type data:, n'oubliez pas que les espaces blancs ont une signification particulière. Vous devez par exemple entrer la chaîne de données sous forme de ligne continue. Les sauts de ligne sont sinon assimilés aux données et il est impossible de décoder l'image.

CSS dans AIR

Adobe AIR 1.0 et les versions ultérieures

WebKit prend en charge plusieurs propriétés CSS étendues. Un grand nombre de ces extensions utilisent le préfixe `-webkit`. Notez que certaines de ces extensions sont d'ordre expérimental et risquent d'être supprimées d'une version future de WebKit. Pour plus d'informations sur la prise en charge par Webkit de CSS et des extensions CSS, voir Safari CSS Reference (disponible en anglais uniquement).

Fonctions de WebKit non prises en charge dans AIR

Adobe AIR 1.0 et les versions ultérieures

AIR ne prend pas en charge les fonctions suivantes disponibles dans WebKit ou Safari 4 :

- Echange de messages inter domaines via window.postMessage (AIR intègre ses propres API de communication inter domaines)
- Variables CSS
- Polices SVG et Web Open Font Format (WOFF)
- Balises vidéo et audio HTML
- Requêtes associées aux périphériques multimédias
- Mémoire cache d'application déconnectée
- Impression (AIR intègre sa propre API PrintJob)
- Vérificateurs orthographiques et grammaticaux
- SVG
- WAI-ARIA
- WebSockets (AIR intègre ses propres API de socket)
- Traitements Web
- API SQL WebKit (AIR intègre sa propre API)
- API de géolocalisation WebKit (AIR intègre sa propre API de géolocalisation sur les périphériques pris en charge)
- API de téléchargement de fichiers multiples WebKit
- Evénements tactiles WebKit (AIR intègre ses propres événements tactiles)
- Langage WML (Wireless Markup Language)

Les listes suivantes recensent des API JavaScript, des éléments HTML, ainsi que des propriétés et valeurs CSS spécifiques non pris en charge par AIR :

Membres d'objet Window JavaScript non pris en charge :

- applicationCache()
- console

- openDatabase()
- postMessage()
- document.print()

Balises HTML non prises en charge :

- audio
- video

Attributs HTML non pris en charge :

- aria-*
- draggable
- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange
- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove
- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

Evénements JavaScript non pris en charge :

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove

- touchend
- touchcancel
- webkitbeginfullscreen
- webkitendfullscreen

Propriétés CSS non prises en charge :

- background-clip
- background-origin (utilisez -webkit-background-origin)
- background-repeat-x
- background-repeat-y
- background-size (utilisez -webkit-background-size)
- border-bottom-left-radius
- border-bottom-right-radius
- border-radius
- border-top-left-radius
- border-top-right-radius
- text-rendering
- -webkit-animation-play-state
- -webkit-background-clip
- -webkit-color-correction
- -webkit-font-smoothing

Valeurs CSS non prises en charge :

- Valeurs de la propriété appearance :
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button
- border-box (background-clip et background-origin)
- contain (background-size)
- content-box (background-clip et background-origin)
- cover (background-size)
- Valeurs de la propriété list :
  - afar
  - amharic
  - amharic-abegede
  - cjk-earthly-branch
  - cjk-heavenly-stem

- ethiopic
- ethiopic-abegede
- ethiopic-abegede-am-et
- ethiopic-abegede-gez
- ethiopic-abegede-ti-er
- ethiopic-abegede-ti-et
- ethiopic-halehame-aa-er
- ethiopic-halehame-aa-et
- ethiopic-halehame-am-et
- ethiopic-halehame-gez
- ethiopic-halehame-om-et
- ethiopic-halehame-sid-et
- ethiopic-halehame-so-et
- ethiopic-halehame-ti-er
- ethiopic-halehame-ti-et
- ethiopic-halehame-tig
- hangul
- hangul-consonant
- lower-norwegian
- oromo
- sidama
- somali
- tigre
- tigrinya-er
- tigrinya-er-abegede
- tigrinya-et
- tigrinya-et-abegede
- upper-greek
- upper-norwegian

- -wap-marquee (propriété display)

Chapitre 59 : Programmation HTML et JavaScript dans AIR

Adobe AIR 1.0 et les versions ultérieures

Diverses rubriques de programmation portent exclusivement sur le développement d'applications Adobe® AIR® avec HTML et JavaScript. Les informations qui suivent sont importantes, que vous programmiez une application AIR à base d'HTML ou bien sur SWF qui exécute HTML et JavaScript à l'aide de la classe HTMLLoader (ou du composant mx:HTML Flex™).

A propos de la classe HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader d'Adobe AIR définit l'objet d'affichage qui peut afficher du contenu HTML dans une application AIR. Les applications basées sur SWF peuvent ajouter un contrôle HTMLLoader à une fenêtre existante ou créer une fenêtre HTML qui contienne automatiquement un objet HTMLLoader avec `HTMLLoader.createRootWindow()`. Il est possible d'accéder à l'objet HTMLLoader via la propriété `window.htmlLoader` de JavaScript au sein de la page HTML chargée.

Chargement du contenu HTML d'une URL

Adobe AIR 1.0 et les versions ultérieures

Le code suivant charge une URL dans un objet HTMLLoader (ajoutez le HTMLLoader en tant qu'enfant de la scène ou tout autre conteneur d'objet d'affichage pour afficher le contenu HTML dans votre application) :

```
import flash.html.HTMLLoader;

var html:HTMLLoader = new HTMLLoader;
html.width = 400;
html.height = 600;
var urlReq:URLRequest = new URLRequest("http://www.adobe.com/");
html.load(urlReq);
```

Les propriétés `width` et `height` d'un objet HTMLLoader sont toutes deux définies sur 0 par défaut. Vous allez vouloir définir ces dimensions lorsque vous ajouterez un objet HTMLLoader à la scène. Cet objet distribue plusieurs événements à mesure qu'une page se charge. Vous pouvez utiliser ces événements pour déterminer à quel moment on peut interagir sans risque avec la page chargée. Ces événements sont décrits dans la section « Gestion des événements HTML dans AIR » à la page 1062.

Remarque : *dans la structure Flex, seules les classes qui étendent la classe UIComponent peuvent être ajoutées en tant qu'enfants de composants d'un conteneur Flex. C'est pour cette raison que vous ne pouvez pas ajouter directement un HTMLLoader en tant qu'enfant d'un composant de conteneur Flex. Toutefois, vous pouvez utiliser le contrôle mx:HTML de Flex, construire une classe personnalisée qui étend l'UIComponent et contient un HTMLLoader comme enfant d'un UIComponent ou bien ajouter le HTMLLoader en tant qu'enfant d'un UIComponent et ajouter l'UIComponent au conteneur Flex.*

Vous pouvez également afficher le texte HTML à l'aide de la classe TextField, mais ses capacités sont limitées. La classe TextField d'Adobe® Flash® Player prend en charge un sous-ensemble du balisage de HTML ; mais, en raison de restrictions de taille, ses capacités sont réduites. La classe HTMLLoader contenue dans Adobe AIR n'est pas disponible dans Flash Player.

Chargement de contenu HTML depuis une chaîne

Adobe AIR 1.0 et les versions ultérieures

La méthode `loadString()` d'un objet HTMLLoader charge une chaîne de contenu HTML dans un objet HTMLLoader :

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

Par défaut, le contenu chargé via la méthode `loadString()` est placé dans un sandbox hors de l'application avec les caractéristiques suivantes :

- Il peut charger le contenu depuis le réseau (mais pas depuis le système de fichiers).
- Il ne peut pas charger les données avec XMLHttpRequest.
- La propriété `window.location` est définie sur `"about:blank"`.
- Le contenu ne peut pas accéder à la propriété `window.runtime` (alors que le contenu placé dans tout sandbox non applicatif le peut).

Dans AIR 1.5, la classe HTMLLoader comprend une propriété `placeLoadStringContentInApplicationSandbox`. Lorsque cette propriété est définie sur `true` pour un objet HTMLLoader, le contenu chargé par la méthode `loadString()` est placé dans le sandbox de l'application. (La valeur par défaut est `false`.) Le contenu chargé par la méthode `loadString()` a ainsi accès à la propriété `window.runtime` et à toutes les API AIR. Si vous définissez cette propriété sur `true`, assurez-vous que la source des données d'une chaîne utilisée dans un appel à la méthode `loadString()` soit approuvée. Les instructions du code de la chaîne HTML sont exécutées avec tous les privilèges de l'application lorsque cette propriété est définie sur `true`. Ne définissez cette propriété sur `true` que si vous êtes certain(e) que la chaîne ne contient que du code inoffensif.

Dans les applications compilées avec les kits SDK d'AIR 1.0 ou AIR 1.1, le contenu chargé par la méthode `loadString()` est placé dans le sandbox de l'application.

Règles de sécurité importantes lors de l'utilisation de contenu HTML dans des applications AIR

Adobe AIR 1.0 et les versions ultérieures

Les fichiers que vous installez avec l'application AIR ont accès aux interfaces de programmation d'AIR. Pour des raisons de sécurité, ce n'est pas le cas pour du contenu provenant d'autres sources. Par exemple, une telle restriction empêche le contenu provenant d'un domaine distant, comme http://example.com, de lire le contenu du répertoire dans l'ordinateur de bureau de l'utilisateur (ou pire encore).

Comme il est possible d'exploiter des failles de sécurité en appelant la fonction `eval()` et les interfaces de programmation qui lui sont liées, l'utilisation de ces méthodes est interdite, par défaut, au contenu installé avec l'application. Toutefois, quelques structures Ajax utilisent la fonction `eval()` et les interfaces de programmation qui lui sont liées.

Pour structurer convenablement le contenu qui va travailler dans une application AIR, il faut tenir compte des règles qui régissent les restrictions liées à la sécurité et qu'il faut appliquer à du contenu provenant de sources différentes. Un tel contenu est placé dans des classifications de sécurité distinctes appelées sandbox de sécurité (voir « Sandbox de sécurité » à la page 1087). Par défaut, un contenu installé avec l'application l'est dans un sandbox appelé sandbox de *l'application*. Il a alors accès aux interfaces de programmation d'AIR. Le sandbox de l'application est en règle générale le sandbox le plus sécurisé, doté de restrictions conçues pour empêcher l'exécution de code non approuvé.

Le moteur d'exécution vous permet de charger le contenu installé avec votre application dans un sandbox autre que celui de l'application. Le contenu dans des sandbox hors application s'exécute dans un environnement similaire à celui d'un navigateur Web classique. Par exemple, le code dans des sandbox hors application peut utiliser `eval()` et des méthodes apparentées. Par contre, il ne peut pas accéder aux interfaces de programmation d'AIR. Le moteur d'exécution contient des dispositifs pour permettre aux contenus de sandbox différents de communiquer entre eux en toute sécurité, sans exposer les interfaces de programmation d'AIR à un contenu hors application, par exemple. Pour plus d'informations, voir la section « Programmation croisée du contenu dans des sandbox de sécurité distincts » à la page 1038.

Si vous appelez du code interdit d'exécution dans un sandbox pour des raisons de sécurité, une erreur JavaScript est signalée par le message : « Violation des règles de sécurité d'Adobe AIR par du code JavaScript dans le sandbox de sécurité de l'application ».

Pour éviter cette erreur, suivez les règles de codage présentées dans la section suivante, « Contournement des erreurs JavaScript liées à la sécurité » à la page 1021.

Pour plus d'informations, voir « Sécurité HTML dans Adobe AIR » à la page 1127.

Contournement des erreurs JavaScript liées à la sécurité

Adobe AIR 1.0 et les versions ultérieures

Si vous appelez du code interdit d'exécution dans un sandbox en raison de ces restrictions liées à la sécurité, le moteur d'exécution distribue une erreur JavaScript : « Violation des règles de sécurité dans le sandbox de sécurité de l'application par le moteur d'exécution d'Adobe AIR sur du code JavaScript ». Pour éviter cette erreur, suivez les règles de codage suivantes.

Causes des erreurs JavaScript liées à la sécurité

Adobe AIR 1.0 et les versions ultérieures

Le code qui s'exécute dans le sandbox de l'application ne peut pas exécuter des opérations qui impliquent l'évaluation et l'exécution de chaînes dès le moment où l'événement `load` est lancé et que les gestionnaires d'événement `load` ne sont plus actifs. Les tentatives d'utilisation des types suivants d'instructions JavaScript, qui évaluent et exécutent des chaînes potentiellement non protégées, génèrent des erreurs JavaScript.

- eval(), fonction
- setTimeout() et setInterval()
- Constructeur Function

De plus, les types d'instructions JavaScript suivants échouent sans générer une erreur JavaScript risquée :

- javascript: URL
- Rappels d'événements affectés par les attributs onevent dans les instructions innerHTML et outerHTML

- Chargement de fichiers JavaScript à partir d'emplacements hors du répertoire d'installation de l'application
- document.write() et document.writeln()
- XMLHttpRequests synchrones lancées avant l'événement load ou au cours de l'exécution d'un gestionnaire d'événement load
- Eléments de script créés dynamiquement

  Remarque : *dans des cas bien particuliers, l'évaluation de chaînes est autorisée. Pour plus d'informations, voir « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130.*

  Adobe gère une liste de structures Ajax reconnues pour leur prise en charge du sandbox de sécurité d'application à l'adresse http://www.adobe.com/go/airappsandboxframeworks_fr.

  Les sections ci-dessous décrivent comment réécrire des scripts pour éviter d'obtenir ces erreurs JavaScript risquées et ces échecs silencieux lors de l'exécution de code dans le sandbox de l'application.

Mappage du contenu de l'application dans un sandbox différent

Adobe AIR 1.0 et les versions ultérieures

Le plus souvent, vous pouvez réécrire ou restructurer une application pour éviter des erreurs JavaScript liées à la sécurité. Toutefois, lorsque la réécriture ou la restructuration s'avèrent impossibles, vous pouvez charger le contenu de l'application dans un sandbox différent à l'aide de la technique décrite à la section « Chargement du contenu de l'application dans un sandbox hors application » à la page 1039. Si ce contenu doit également accéder aux interfaces de programmation d'AIR, vous pouvez créer un pont de sandbox, tel que décrit à la section « Configuration d'une interface de pont de sandbox » à la page 1040.

eval(), fonction

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans le sandbox de l'application, la fonction `eval()` ne peut être utilisée qu'avant l'exécution de l'événement `load` de page ou qu'en cours d'exécution d'un gestionnaire d'événement `load`. Après le chargement de la page, les appels à `eval()` n'exécuteront pas de code. Toutefois, dans les situations suivantes, il vous est possible de réécrire le code pour éviter d'utiliser `eval()`.

Affectation des propriétés à un objet

Adobe AIR 1.0 et les versions ultérieures

Au lieu d'analyser une chaîne pour construire l'accesseur propriété :

```
eval("obj." + propName + " = " + val);
```

accédez aux propriétés par une notation crochets :

```
obj[propName] = val;
```

Création d'une fonction avec des variables disponibles dans le contexte

Adobe AIR 1.0 et les versions ultérieures

Remplacez les instructions telles que les suivantes :

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

par :

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

Création d'un objet à l'aide du nom de la classe comme paramètre de chaîne

Adobe AIR 1.0 et les versions ultérieures

Considérez une classe de JavaScript hypothétique définie de la manière suivante :

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

La façon la plus simple de créer une occurrence consisterait à utiliser `eval()` :

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

Toutefois, vous pouvez éviter l'appel à `eval()` en analysant chaque composant du nom de la classe et en construisant le nouvel objet à l'aide de la notation crochets :

```
function getter(str)
{
    var obj = window;
    var names = str.split('.');
    for(var i=0;i<names.length;i++){
        if(typeof obj[names[i]]=='undefined'){
            var undefstring = names[0];
            for(var j=1;j<=i;j++)
                undefstring+="."+names[j];
            throw new Error(undefstring+" is undefined");
        }
        obj = obj[names[i]];
    }
    return obj;
}
```

Pour créer l'occurrence, utilisez :

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() et setInterval()

Adobe AIR 1.0 et les versions ultérieures

Remplacez la chaîne passée en tant que fonction gestionnaire avec une référence de fonction ou un objet de fonction. Par exemple, remplacez une instruction telle que :

```
setTimeout("alert('Timeout')", 100);
```

par :

```
setTimeout(function(){alert('Timeout')}, 100);
```

Ou bien, lorsque la fonction demande que l'objet `this` soit paramétré par l'appelant, remplacez une instruction telle que :

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

par ce qui suit :

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

Constructeur Function

Adobe AIR 1.0 et les versions ultérieures

Les appels à `new Function(param, body)` peuvent être remplacés par une déclaration de fonction inline ou utilisés seulement avant que l'événement `load` de la page n'ait été traité.

javascript: URL

Adobe AIR 1.0 et les versions ultérieures

Le code défini dans un lien à l'aide du modèle javascript: URL est ignoré dans le sandbox de l'application. Aucune erreur JavaScript risquée n'est générée. Vous pouvez remplacer des liens à l'aide d'URL JavaScript: telles que :

```
<a href="javascript:code()">Click Me</a>
```

par :

```
<a href="#" onclick="code()">Click Me</a>
```

Rappels d'événements affectés par les attributs onevent dans les instructions innerHTML et outerHTML

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous utilisez innerHTML ou outerHTML pour ajouter des éléments au DOM d'un document, tous les rappels d'événements affectés au sein de l'instruction, tels que `onclick` ou `onmouseover` sont ignorés. Aucun message d'erreur sur la sécurité n'est généré. A la place, vous pouvez affecter un attribut `id` aux nouveaux éléments et paramétrer les fonctions de rappel de gestionnaires d'événement à l'aide de la méthode `addEventListener()`.

Par exemple, étant donné un élément-cible dans un document tel que :

```
<div id="container"></div>
```

Remplacez les instructions telles que :

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

par :

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

Chargement de fichiers JavaScript à partir d'emplacements hors du répertoire d'installation de l'application

Adobe AIR 1.0 et les versions ultérieures

Le chargement de fichiers de script à partir d'emplacements hors du sandbox de l'application n'est pas autorisé. Aucun message d'erreur sur la sécurité n'est généré. Tous les fichiers de script qui s'exécutent dans le sandbox de l'application doivent être installés dans le répertoire de l'application. Pour utiliser des scripts externes dans une page, il vous faut mapper la page dans un sandbox distinct. Voir la section « Chargement du contenu de l'application dans un sandbox hors application » à la page 1039.

document.write() et document.writeln()

Adobe AIR 1.0 et les versions ultérieures

Les appels à `document.write()` ou `document.writeln()` sont ignorés après le traitement de l'événement `load` de la page. Aucun message d'erreur sur la sécurité n'est généré. Vous pouvez également charger un nouveau fichier ou remplacer le corps du document à l'aide des techniques de manipulation du DOM :

XMLHttpRequests synchrones lancées avant l'événement load ou au cours de l'exécution d'un gestionnaire d'événement load

Adobe AIR 1.0 et les versions ultérieures

Les XMLHttpRequests synchrones lancées avant l'événement `load` de la page ou au cours du traitement d'un événement `load` ne renvoient pas de contenu. Les XMLHttpRequests asynchrones peuvent être lancées, mais elles ne renvoient rien avant la fin de l'événement`load`. Après le traitement de l'événement `load`, les XMLHttpRequests synchrones se comportent normalement.

Eléments de script créés dynamiquement

Adobe AIR 1.0 et les versions ultérieures

Les éléments de script créés dynamiquement, tels ceux créés avec les méthodes innerHTML ou `document.createElement()` sont ignorés.

Accès aux classes de l'API AIR à partir de JavaScript

Adobe AIR 1.0 et les versions ultérieures

En plus des éléments standard et étendus du Webkit, HTML et le code JavaScript peuvent accéder aux classes hôtes par le moteur d'exécution. Ces classes vous permettent d'accéder aux fonctions avancées fournies par AIR et notamment :

- Accès au système de fichiers
- Utilisation des bases de données SQL locales
- Contrôle des menus Fenêtre et de l'application
- Accès aux sockets pour la mise en réseau
- Utilisation des classes et objets définis par l'utilisateur
- Capacités de sonorisation

  Par exemple, l'API de fichiers d'AIR comprend une classe File contenue dans le package flash.filesystem Vous pouvez créer un objet File en JavaScript comme suit :

  ```
  var myFile = new window.runtime.flash.filesystem.File();
  ```

  L'objet `runtime` est un objet JavaScript spécial, mis à la disposition du contenu HTML s'exécutant dans AIR, dans le sandbox de l'application. Il vous permet d'accéder aux classes runtime à partir de JavaScript. La propriété `flash` de l'objet `runtime` fournit un accès au package Flash. A son tour, la propriété `flash.filesystem` de l'objet `runtime` fournit un accès au package flash.filesystem. Celui-ci contient la classe File. Les packages constituent un moyen d'organiser les classes utilisées dans ActionScript.

  Remarque : *la propriété `runtime` n'est pas automatiquement ajoutée aux objets window des pages chargées dans une image ou dans une iframe. Toutefois, tant que le document enfant se trouve dans le sandbox de l'application, l'enfant peut accéder à la propriété `runtime` du parent.*

  La structure du package des classes runtime demanderait aux développeurs de taper de longues chaînes de code JavaScript pour accéder à chaque classe, comme dans `window.runtime.flash.desktop.NativeApplication`. C'est pourquoi le SDK d'AIR contient un fichier AIRAliases.js qui vous permet d'accéder à des classes runtime beaucoup plus facilement, par exemple en tapant tout simplement `air.NativeApplication`.

  Les classes de l'interface de programmation AIR sont abordées tout au long du présent guide. D'autres classes de l'interface de programmation Flash Player, qui peuvent intéresser les développeurs HTML, sont décrites dans le manuel *Adobe AIR API Reference for HTML Developers* (disponible en anglais uniquement). ActionScript est le langage utilisé dans le contenu SWF (Flash Player). Toutefois, la syntaxe de JavaScript et celle d'ActionScript sont très proches. Elles sont toutes deux basées sur des versions du langage ECMAScript. Toutes les classes intégrées sont disponibles à la fois en JavaScript (dans un contenu HTML) et en ActionScript (dans un contenu SWF).

  Remarque : *le code JavaScript ne peut utiliser les classes Dictionary, XML et XMLList qui sont disponibles dans ActionScript.*

Utilisation du fichier AIRAliases.js

Adobe AIR 1.0 et les versions ultérieures

Les classes d'exécution sont organisées en une structure de package, comme suit :

- `window.runtime.flash.desktop.NativeApplication`
- `window.runtime.flash.desktop.ClipboardManager`
- `window.runtime.flash.filesystem.FileStream`
- `window.runtime.flash.data.SQLDatabase`

Dans le SDK d'AIR se trouve un fichier AIRAliases.js qui fournit des définitions d'« alias » vous permettant d'accéder à des classes d'exécution avec moins de frappes au clavier. Par exemple, vous pouvez accéder aux classes répertoriées ci-dessus en tapant tout simplement ce qui suit :

- `air.NativeApplication`
- `air.Clipboard`
- `air.FileStream`
- `air.SQLDatabase`

Cette liste est simplement un petit sous-ensemble des classes du fichier AIRAliases.js. La liste complète des classes et des fonctions de niveau package figure dans le manuel *Adobe AIR API Reference for HTML Developers* (disponible en anglais uniquement).

En plus des classes d'exécution utilisées le plus souvent, le fichier AIRAliases.js contient des alias pour les fonctions les plus courantes au niveau du package : `window.runtime.trace()`, `window.runtime.flash.net.navigateToURL()` et `window.runtime.flash.net.sendToURL()` qui ont pour alias `air.trace()`, `air.navigateToURL()` et `air.sendToURL()`.

Pour utiliser le fichier AIRAliases.js, insérez la référence `script` suivante dans votre page HTML :

```
<script src="AIRAliases.js"></script>
```

Ajustez la page dans la référence `src` selon les besoins.

Important : *sauf indication contraire, le code JavaScript présenté dans cette documentation à titre d'exemple suppose que vous avez inclus le fichier AIRAliases.js dans votre page HTML.*

A propos des URL dans AIR

Adobe AIR 1.0 et les versions ultérieures

Dans un contenu HTML qui s'exécute sous AIR, vous pouvez utiliser les modèles d'URL suivants lorsque vous définissez les attributs `src` pour les balises `img`, `frame`, `iframe` et `script`, dans l'attribut `href` d'une balise `link` ou dans tout autre emplacement où vous pouvez fournir une URL.

Modèle d'URL	Description	Exemple
file	Un chemin relatif à la racine du système de fichiers	`file:///c:/AIR Test/test.txt`
app	Un chemin relatif au répertoire racine de l'application installée.	`app:/images`
app-storage	Un chemin relatif au répertoire de stockage de l'application Pour chaque application installée, AIR définit un répertoire de stockage de l'application qui est unique. C'est un emplacement pratique pour y enregistrer des données spécifiques à cette application.	`app-storage:/settings/prefs.xml`
http	Une requête HTTP standard	`http://www.adobe.com`
https	Une requête HTTPS standard	`https://secure.example.com`

Pour plus d'informations sur l'utilisation des modèles d'URL d'AIR, voir la section « Modèles d'URI » à la page 845.

De nombreuses interfaces de programmation AIR, y compris les classes File, Loader, URLStream et Sound, utilisent un objet URLRequest plutôt qu'une chaîne contenant l'URL. L'objet URLRequest lui-même est initialisé avec une chaîne qui peut utiliser l'un des mêmes modèles d'URL. Par exemple, l'instruction suivante crée un objet URLRequest qui peut être utilisé pour demander la page d'accueil d'Adobe :

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

Pour des informations sur les objets URLRequest, voir la section « Communications HTTP » à la page 843.

Mise des objets ActionScript à disposition de JavaScript

Adobe AIR 1.0 et les versions ultérieures

Le JavaScript dans la page HTML chargée par un objet HTMLLoader peut appeler les classes, les objets et les fonctions définis dans le contexte d'exécution d'ActionScript à l'aide des propriétés `window.runtime`, `window.htmlLoader` et `window.nativeWindow` de la page HTML. Par la création de références aux objets et fonctions ActionScript au sein d'un contexte d'exécution de JavaScript, vous pouvez également mettre ceux-ci à la disposition du code JavaScript.

Exemple de base pour l'accès à des objets JavaScript à partir d'ActionScript

Adobe AIR 1.0 et les versions ultérieures

L'exemple ci-dessous décrit comment ajouter des propriétés qui font référence à des objets dans l'objet window global d'une page HTML.

```
var html:HTMLLoader = new HTMLLoader();
var foo:String = "Hello from container SWF."
function helloFromJS(message:String):void {
    trace("JavaScript says:", message);
}
var urlReq:URLRequest = new URLRequest("test.html");
html.addEventListener(Event.COMPLETE, loaded);
html.load(urlReq);

function loaded(e:Event):void{
    html.window.foo = foo;
    html.window.helloFromJS = helloFromJS;
}
```

Le contenu HTML, qui se trouve dans un fichier appelé test.html, chargé dans un objet HTMLLoader de l'exemple précédent, peut accéder à la propriété `foo` et à la méthode `helloFromJS()` définies dans le fichier SWF parent :

```
<html>
    <script>
        function alertFoo() {
            alert(foo);
        }
    </script>
    <body>
        <button onClick="alertFoo()">
            What is foo?
        </button>
        <p><button onClick="helloFromJS('Hi.')">
            Call helloFromJS() function.
        </button></p>
    </body>
</html>
```

Lorsque vous accédez au contexte JavaScript d'un document en cours de chargement, vous pouvez utiliser l'événement `htmlDOMInitialize` pour créer des objets suffisamment tôt dans la séquence de création de la page pour que des scripts définis dans la page puissent y accéder. Si vous attendez l'événement `complete`, seuls les scripts de la page qui s'exécutent après l'événement `load` de la page peuvent accéder aux objets ajoutés.

Mise des définitions de classe à disposition de JavaScript

Adobe AIR 1.0 et les versions ultérieures

Pour mettre les classes ActionScript de votre application à la disposition de JavaScript, vous pouvez affecter le contenu HTML chargé au domaine de l'application qui contient les définition de classes. Le domaine d'application du contexte d'exécution de JavaScript peut être défini avec la propriété `runtimeApplicationDomain` de l'objet HTMLLoader. Pour définir le domaine d'application sur le domaine d'application principal, par exemple, définissez `runtimeApplicationDomain` sur `ApplicationDomain.currentDomain`, comme le montre le code ci-dessous :

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

Dès que la propriété `runtimeApplicationDomain` est définie, le contexte JavaScript partage les définitions de classe avec le domaine affecté. Pour créer une occurrence d'une classe personnalisée dans JavaScript, faites référence à la définition de classe par la propriété `window.runtime` et utilisez l'opérateur `new`.

```
var customClassObject = new window.runtime.CustomClass();
```

Le contenu HTML doit provenir d'un domaine de sécurité compatible. S'il provient d'un domaine de sécurité autre que celui de l'application auquel vous l'affectez, la page utilise plutôt un domaine d'application par défaut. Par exemple, si vous chargez à partir d'Internet une page distante, vous ne pourriez pas affecter ApplicationDomain.currentDomain comme domaine d'application de la page.

Suppression des écouteurs d'événement

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous ajoutez des écouteurs d'événement à des objets hors de la page en cours, y compris des objets d'exécution, des objets dans du contenu SWF chargé et même des objets JavaScript qui s'exécutent dans d'autres pages, vous devriez toujours supprimer les écouteurs d'événement lorsque la page décharge. Autrement, l'écouteur d'événement distribue l'événement à une fonction de gestionnaire qui n'existe plus. Si cela se produit, le message d'erreur suivant s'affiche : « The application attempted to reference a JavaScript object in an HTML page that is no longer loaded » (L'application a tenté de référencer un objet JavaScript sur une page HTML qui n'est plus chargée). La suppression d'écouteurs d'événement qui ne sont plus utiles permet aussi à AIR de récupérer la mémoire qui leur est associée. Pour plus d'informations, voir la section « Suppression d'écouteurs d'événement sur une page HTML de navigation » à la page 1067.

Accès au DOM HTML et aux objets JavaScript à partir d'ActionScript

Adobe AIR 1.0 et les versions ultérieures

Dès que l'objet HTMLLoader distribue l'événement `complete`, vous pouvez accéder à tous les objets dans le DOM (ou document object model) HTML pour la page. Les objets accessibles sont les éléments d'affichage, tels que les objets `div` et `p` dans la page, ainsi que les variables et fonctions JavaScript. L'événement `complete` correspond à l'événement `load` de la page JavaScript. Avant que `complete` ne soit distribué, les éléments DOM, les variables et les fonctions peuvent ne pas avoir été analysés ou créés. Si possible, attendez l'événement `complete` avant d'accéder au DOM HTML.

Par exemple, examinez la page HTML suivante :

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

La page HTML simple définit une variable JavaScript appelée *foo* et une fonction JavaScript appelée *test()*. Ces deux éléments sont des propriétés de l'objet `window` global de la page. En outre, l'objet `window.document` comprend un élément appelé P, pourvu de l'ID *p1*), auquel vous pouvez accéder à l'aide de la méthode `getElementById()`. Dès que la page est chargée, c'est-à-dire lorsque l'objet HTMLLoader distribue l'événement `complete`, vous pouvez accéder à chacun de ces objets à partir d'ActionScript, comme le montre le code ActionScript ci-dessous :

```
 var html:HTMLLoader = new HTMLLoader();
html.width = 300;
html.height = 300;
html.addEventListener(Event.COMPLETE, completeHandler);
var xhtml:XML =
    <html>
        <script>
            foo = 333;
            function test() {
                return "OK.";
            }
        </script>
        <body>
            <p id="p1">Hi.</p>
        </body>
    </html>;
html.loadString(xhtml.toString());

function completeHandler(e:Event):void {
    trace(html.window.foo); // 333
    trace(html.window.document.getElementById("p1").innerHTML); // Hi.
    trace(html.window.test()); // OK.
}
```

Pour accéder au contenu d'un élément HTML, utilisez la propriété `innerHTML`. Par exemple, le code ci-dessus utilise `html.window.document.getElementById("p1").innerHTML` pour lire le contenu de l'élément HTML appelé *p1*.

Vous pouvez également paramétrer les propriétés de la page HTML à partir d'ActionScript. Ainsi, l'exemple ci-dessous définit le contenu de l'élément `p1` et la valeur de la variable JavaScript `foo` sur la page à l'aide d'une référence à l'objet conteneur HTMLLoader :

```
 html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

Intégration d'un contenu SWF en HTML

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez intégrer un contenu SWF dans un contenu HTML au sein d'une application AIR comme vous le feriez dans un navigateur. Intégrez le contenu SWF à l'aide d'une balise `object`, d'une balise `embed` ou de toutes les deux.

Remarque : *une pratique courante de développement sur le Web consiste à utiliser à la fois une balise `object` et une balise `embed` pour afficher un contenu SWF dans une page HTML. Vous ne tirez aucun avantage de cette pratique avec AIR. Vous pouvez utilisez la seule balise `object`, conforme à la norme W3C, dans le contenu pour l'afficher dans AIR. Cependant, vous pouvez continuer à utiliser les balises `object` et `embed` ensemble, le cas échéant, pour du contenu HTML qui est également affiché dans un navigateur.*

Si vous avez activé la transparence dans l'objet NativeWindow qui affiche le contenu HTML et SWF, AIR n'affiche pas le contenu SWF lorsque le mode Fenêtre (`wmode`) activé pour intégrer le contenu est défini sur la valeur `window`. Pour afficher le contenu SWF dans une page HTML de fenêtre transparente, définissez le paramètre `wmode` sur `opaque` ou `transparent`. Etant donné que le paramètre `window` correspond à la valeur par défaut de `wmode`, le contenu risque de ne pas être affiché si vous ne stipulez pas de valeur.

L'exemple ci-dessous illustre l'utilisation de la balise `object` HTML pour afficher un fichier SWF au sein d'un contenu HTML. Le paramètre `wmode` étant défini sur `opaque`, le contenu est affiché même si l'objet NativeWindow sous-jacent est transparent. Le fichier SWF est chargé à partir du répertoire de l'application, mais vous pouvez utiliser tout modèle d'URL pris en charge par AIR. L'emplacement à partir duquel le fichier SWF est chargé détermine le sandbox de sécurité dans lequel AIR place le contenu.

```
<object type="application/x-shockwave-flash" width="100%" height="100%">
    <param name="movie" value="app:/SWFFile.swf"></param>
    <param name="wmode" value="opaque"></param>
</object>
```

Vous pouvez également utiliser un script pour charger un contenu dynamiquement. L'exemple ci-dessous crée un nœud `object` pour afficher le fichier SWF spécifié dans le paramètre `urlString`. L'exemple ajoute le nœud en tant qu'enfant de l'élément de page avec l'ID spécifiée par le paramètre `elementID`.

```
<script>
function showSWF(urlString, elementID){
    var displayContainer = document.getElementById(elementID);
    var flash = createSWFObject(urlString, 'opaque', 650, 650);
    displayContainer.appendChild(flash);
}
function createSWFObject(urlString, wmodeString, width, height){
    var SWFObject = document.createElement("object");
    SWFObject.setAttribute("type","application/x-shockwave-flash");
    SWFObject.setAttribute("width","100%");
    SWFObject.setAttribute("height","100%");
    var movieParam = document.createElement("param");
    movieParam.setAttribute("name","movie");
    movieParam.setAttribute("value",urlString);
    SWFObject.appendChild(movieParam);
    var wmodeParam = document.createElement("param");
    wmodeParam.setAttribute("name","wmode");
    wmodeParam.setAttribute("value",wmodeString);
    SWFObject.appendChild(wmodeParam);
    return SWFObject;
}
</script>
```

Le contenu SWF n'est pas affiché si l'objet HTMLLoader est mis à l'échelle ou pivoté, ou si la propriété `alpha` est définie sur une valeur autre que 1.0. Dans les versions d'AIR antérieures à 1.5.2, le contenu SWF ne s'affichait pas dans une fenêtre transparente, quelle que soit la valeur définie de `wmode`.

Remarque : *lorsqu'un objet SWF incorporé tente de charger un actif externe, notamment un fichier vidéo, il est possible que le contenu SWF ne soit pas rendu correctement si un chemin absolu vers le fichier vidéo n'est pas fourni dans le fichier HTML. Un objet SWF incorporé peut néanmoins charger un fichier image externe à l'aide d'un chemin relatif.*

L'exemple suivant explique comment charger les actifs externes via un objet SWF incorporé dans un contenu HTML :

```
var imageLoader;

function showSWF(urlString, elementID){
    var displayContainer = document.getElementById(elementID);
    imageLoader = createSWFObject(urlString,650,650);
    displayContainer.appendChild(imageLoader);
}

function createSWFObject(urlString, width, height){

    var SWFObject = document.createElement("object");
    SWFObject.setAttribute("type","application/x-shockwave-flash");
    SWFObject.setAttribute("width","100%");
    SWFObject.setAttribute("height","100%");

    var movieParam = document.createElement("param");
    movieParam.setAttribute("name","movie");
    movieParam.setAttribute("value",urlString);
    SWFObject.appendChild(movieParam);

    var flashVars = document.createElement("param");
    flashVars.setAttribute("name","FlashVars");

    //Load the asset inside the SWF content.
    flashVars.setAttribute("value","imgPath=air.jpg");
    SWFObject.appendChild(flashVars);

    return SWFObject;
}
function loadImage()
{
  showSWF("ImageLoader.swf", "imageSpot");

}
```

Dans l'exemple ActionScript suivant, le chemin d'image transmis par le fichier HTML est lu et l'image est chargée sur la scène :

```
package
{
      import flash.display.Sprite;
      import flash.display.LoaderInfo;
      import flash.display.StageScaleMode;
      import flash.display.StageAlign;
      import flash.display.Loader;
      import flash.net.URLRequest;

      public class ImageLoader extends Sprite
      {
            public function ImageLoader()
            {

                  var flashvars = LoaderInfo(this.loaderInfo).parameters;

                  if(flashvars.imgPath){
                        var imageLoader = new Loader();
                        var image = new URLRequest(flashvars.imgPath);
                        imageLoader.load(image);
                        addChild(imageLoader);
                        imageLoader.x = 0;
                        imageLoader.y = 0;
                        stage.scaleMode=StageScaleMode.NO_SCALE;
                        stage.align=StageAlign.TOP_LEFT;
                  }
            }
      }
}
```

Utilisation des bibliothèques ActionScript au sein d'une page HTML

Adobe AIR 1.0 et les versions ultérieures

AIR étend l'élément de script HTML de telle sorte qu'une page peut importer des classes ActionScript dans un fichier SWF compilé. Par exemple, pour importer une bibliothèque appelée *myClasses.swf* qui se trouve dans le sous-répertoire `lib` du dossier racine de l'application, insérez la balise de script suivante au sein d'un fichier HTML :

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

Important : *l'attribut type doit être `type="application/x-shockwave-flash"` pour que la bibliothèque soit chargée convenablement.*

Si le contenu SWF est compilé sous forme de fichier SWF Flash Player 10 ou AIR 1.5, vous devez définir l'espace de noms XML du fichier descripteur d'application sur l'espace de noms AIR 1.5.

Le répertoire `lib` et le fichier `myClasses.swf` doivent aussi être inclus lorsque le fichier AIR est mis en package.

Accédez aux classes importées par la propriété `runtime` de l'objet Window de JavaScript :

```
var libraryObject = new window.runtime.LibraryClass();
```

Si les classes du fichier SWF sont organisées en packages, vous devez également inclure le nom du package. Par exemple, si la définition LibraryClass se trouvait dans un package appelé *utilities*, vous créeriez une occurrence de la classe à l'aide de l'instruction suivante :

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

Remarque : *pour compiler une bibliothèque de SWF dans ActionScript afin qu'elle fasse partie d'une page HTML dans AIR, utilisez le compilateur* `acompc`*. L'utilitaire acompc fait partie du SDK de Flex. Il est décrit dans la documentation SDK de Flex.*

Accès au DOM HTML et aux objets JavaScript à partir d'un fichier ActionScript importé

Adobe AIR 1.0 et les versions ultérieures

Pour accéder à des objets dans une page HTML à partir d'ActionScript dans un fichier SWF importé dans la page à l'aide de la balise `<script>`, transmettez une référence à l'objet JavaScript, telle que `window` ou `document`, à une fonction définie dans le code ActionScript. Utilisez la référence au sein de la fonction pour accéder à l'objet ou à d'autres objets accessibles grâce à la référence qu'on a fait passer.

Par exemple, examinez la page HTML suivante :

```
<html>
    <script src="ASLibrary.swf" type="application/x-shockwave-flash"></script>
    <script>
        num = 254;
        function getStatus() {
            return "OK.";
        }
        function runASFunction(window){
            var obj = new runtime.ASClass();
            obj.accessDOM(window);
        }
    </script>
    <body onload="runASFunction">
        <p id="p1">Body text.</p>
    </body>
</html>
```

La page HTML simple contient une variable JavaScript appelée *num* et une fonction JavaScript appelée *getStatus()*. Ces deux éléments sont des propriétés de l'objet `window` de la page. En outre, l'objet `window.document` contient un élément p désigné et dont l'ID est *p1*.

La page charge un fichier ActionScript, « ASLibrary.swf », qui contient une classe, ASClass. ASClass définit une fonction appelée `accessDOM()` qui enregistre simplement les valeurs de ces objets JavaScript. La méthode `accessDOM()` prend l'objet Window de JavaScript comme argument. A l'aide de cette référence Window, elle peut accéder à d'autres objets de la page tels que les variables, les fonctions et les éléments DOM, comme l'illustre la définition ci-dessous :

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

Vous pouvez à la fois lire et définir des propriétés de la page HTML à partir d'une classe ActionScript importée. Par exemple, la fonction suivante définit sur la page le contenu de l'élément `p1` et la valeur de la variable JavaScript `foo` :

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Conversion des objets Date et RegExp

Adobe AIR 1.0 et les versions ultérieures

Les langages JavaScript et ActionScript définissent tous deux les classes Date et RegExp, mais les objets de ces types ne sont pas automatiquement convertis d'un contexte d'exécution à l'autre. Vous devez d'abord convertir les objets Date et RegExp au type équivalent avant de les utiliser pour définir des propriétés ou des paramètres de fonction dans l'autre contexte d'exécution.

Par exemple, le code ActionScript ci-dessous convertit un objet Date de JavaScript appelé `jsDate` en un objet Date ActionScript :

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

Par exemple, le code ActionScript ci-dessous convertit un objet RegExp de JavaScript appelé `jsRegExp` en un objet RegExp d'ActionScript :

```
var flags:String = "";
if (jsRegExp.dotAll) flags += "s";
if (jsRegExp.extended) flags += "x";
if (jsRegExp.global) flags += "g";
if (jsRegExp.ignoreCase) flags += "i";
if (jsRegExp.multiline) flags += "m";
var asRegExp:RegExp = new RegExp(jsRegExp.source, flags);
```

Manipulation d'une feuille de style HTML à partir d'ActionScript

Adobe AIR 1.0 et les versions ultérieures

Dès que l'objet HTMLLoader a distribué l'événement `complete`, vous pouvez étudier et manipuler les styles CSS dans une page.

Par exemple, examinez la page HTML simple suivante :

```
<html>
<style>
    .style1A { font-family:Arial; font-size:12px }
    .style1B { font-family:Arial; font-size:24px }
</style>
<style>
    .style2 { font-family:Arial; font-size:12px }
</style>
<body>
    <p class="style1A">
        Style 1A
    </p>
    <p class="style1B">
        Style 1B
    </p>
    <p class="style2">
        Style 2
    </p>
</body>
</html>
```

Après le chargement de ce contenu par un objet HTMLLoader, vous pouvez manipuler les styles CSS dans la page par le tableau `cssRules` du tableau `window.document.styleSheets`, comme vous pouvez le voir ci-dessous :

```
var html:HTMLLoader = new HTMLLoader( );
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.addEventListener(Event.COMPLETE, completeHandler);
function completeHandler(event:Event):void {
    var styleSheet0:Object = html.window.document.styleSheets[0];
    styleSheet0.cssRules[0].style.fontSize = "32px";
    styleSheet0.cssRules[1].style.color = "#FF0000";
    var styleSheet1:Object = html.window.document.styleSheets[1];
    styleSheet1.cssRules[0].style.color = "blue";
    styleSheet1.cssRules[0].style.font-family = "Monaco";
}
```

Le code ajuste les styles CSS pour que le document HML résultant apparaisse comme ci-dessous :

Style 1A

Style 1B

Style 2

N'oubliez pas que le code ne peut ajouter des styles à la page qu'après la distribution de l'événement `complete` par l'objet HTMLLoader.

Programmation croisée du contenu dans des sandbox de sécurité distincts

Adobe AIR 1.0 et les versions ultérieures

Le modèle de sécurité du moteur d'exécution isole le code de différentes origines. La programmation croisée du contenu dans des sandbox de sécurité distincts permet au contenu d'un sandbox de sécurité d'accéder à des propriétés et à des méthodes sélectionnées dans un autre.

Sandbox de sécurité AIR et code JavaScript

Adobe AIR 1.0 et les versions ultérieures

AIR applique une politique d'origine commune qui empêche le code d'un domaine d'interagir avec le contenu d'un autre. Tous les fichiers sont placés dans un sandbox sur la base de leur origine. Habituellement, le contenu d'un sandbox d'application ne peut pas enfreindre le principe d'origine commune et inter-coder le contenu chargé depuis un emplacement qui se trouve hors du répertoire d'installation de l'application. Toutefois, AIR met à votre disposition quelques techniques qui vous permettent d'inter-coder du contenu hors application.

L'une d'entre elles utilise des images ou des iframes pour mapper le contenu de l'application dans un sandbox de sécurité distinct. Toute page chargée à partir de la zone à sandbox de l'application se comporte comme si elle l'avait été à partir du domaine distant. Par exemple, le mappage du contenu de l'application sur le domaine *example.com* lui permettrait d'inter-coder des pages chargées à partir d'example.com.

Comme cette technique place le contenu de l'application dans un sandbox distinct, le code au sein de ce contenu n'est non plus assujetti aux restrictions qui entourent l'exécution du code des chaînes évaluées. Vous pouvez utiliser la technique de mappage des sandbox pour assouplir ces restrictions, même s'il n'est pas nécessaire d'inter-coder un contenu distant. Le mappage de contenu de cette façon est particulièrement utile lorsque vous travaillez avec l'une des nombreuses structures de JavaScript ou avec du code existant qui se fonde sur l'évaluation des chaînes. Toutefois, vous devriez tenir compte d'un risque supplémentaire et vous en prémunir : celui d'un contenu non approuvé qui pourrait être introduit et exécuté lorsque le contenu tourne hors du sandbox de l'application.

Cependant, un contenu de l'application mappé sur un autre sandbox perd son droit d'accès à des interfaces de programmation d'AIR de telle sorte que la technique de mappage du sandbox ne peut pas être utilisée pour présenter des fonctionnalités d'AIR à du code exécuté hors du sandbox de l'application.

Une autre technique de programmation croisée vous permet de créer une interface, appelée *pont de sandbox*, qui sert de passerelle entre le contenu d'un sandbox hors application et son document parent dans le sandbox de l'application. Le pont permet au contenu enfant d'accéder à des propriétés et à des méthodes définies par le parent et inversement, ou bien les deux à la fois.

Enfin, vous pouvez lancer des XMLHttpRequests entre les domaines à partir du sandbox de l'application et, facultativement, d'autres sandbox.

Pour plus d'informations, voir « Eléments image et iframe HTML » à la page 1011, « Sécurité HTML dans Adobe AIR » à la page 1127 et « Objet XMLHttpRequest » à la page 1005.

Chargement du contenu de l'application dans un sandbox hors application

Adobe AIR 1.0 et les versions ultérieures

Pour permettre à un contenu de l'application d'inter-coder sans risque un contenu chargé à partir d'un emplacement se trouvant hors du répertoire d'installation de l'application, vous pouvez utiliser l'élément `image` ou `iframe` pour charger ce contenu en tant que contenu externe dans le même sandbox de sécurité. Si vous ne voulez pas inter-coder de contenu distant mais que vous souhaitez charger une page de votre application hors du sandbox de l'application, vous pouvez utiliser la même technique, c'est-à-dire spécifier `http://localhost/` ou toute autre valeur inoffensive comme domaine d'origine.

AIR ajoute les nouveaux attributs, `sandboxRoot` et `documentRoot` à l'élément d'image pour vous permettre de spécifier si un fichier d'application chargé dans l'image devrait être mappé sur un sandbox hors application. Les fichiers dont la résolution aboutit à un chemin sous l'URL `sandboxRoot` sont plutôt chargés à partir du répertoire `documentRoot`. Pour des raisons de sécurité, le contenu de l'application chargé ainsi est traité comme s'il avait été chargé à partir de l'URL de `sandboxRoot`.

La propriété `sandboxRoot` spécifie l'URL à utiliser pour déterminer le sandbox et le domaine dans lesquels placer le contenu de l'image. Les modèles d'URL `file:`, `http:` ou `https:` doivent être utilisés. Si vous spécifiez une URL relative, le contenu reste dans le sandbox de l'application.

La propriété `documentRoot` spécifie le répertoire à partir duquel il faut charger le contenu de l'image. Les modèles d'URL `app:`, `http:` ou `app-storage:` doivent être utilisés.

L'exemple ci-dessous mappe le contenu installé dans le sous-répertoire `sandbox` de l'application qui doit s'exécuter dans le sandbox distant et le domaine `www.example.com` :

```
 <iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

La page `ui.html` pourrait charger un fichier JavaScript à partir du dossier `sandbox` local à l'aide de la balise de script suivante :

```
<script src="http://www.example.com/local/ui.js"></script>
```

Elle pourrait également charger un contenu à partir d'un répertoire sur le serveur distant à l'aide d'une balise de script telle que la suivante :

```
<script src="http://www.example.com/remote/remote.js"></script>
```

L'URL `sandboxRoot` masquera tout contenu sur la même URL du serveur distant. Dans l'exemple ci-dessus, vous ne pourriez pas avoir accès à un contenu distant, quel qu'il soit, à l'adresse `www.example.com/local/` ou même à l'un de ses sous-répertoires, car AIR remappe la requête sur le répertoire local de l'application. Les requêtes sont remappées, qu'elles proviennent d'une navigation de pages, d'une XMLHttpRequest ou de tout autre moyen de chargement de contenu.

Configuration d'une interface de pont de sandbox

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez utiliser un pont de sandbox lorsqu'un contenu dans le sandbox de l'application doit accéder à des propriétés ou des méthodes définies par un contenu dans un sandbox hors application ou encore lorsqu'un contenu hors application doit accéder à des propriétés et des méthodes définies par un contenu dans le sandbox de l'application. Créez un pont avec les propriétés `childSandboxBridge` et `parentSandboxBridge` de l'objet `window` de tout document enfant.

Création d'un pont de sandbox enfant

Adobe AIR 1.0 et les versions ultérieures

La propriété `childSandboxBridge` permet au document enfant de présenter une interface au contenu dans le document parent. Pour présenter une interface, vous définissez la propriété `childSandbox` sur une fonction ou un objet dans le document enfant. Vous pouvez alors accéder à l'objet ou la fonction à partir du contenu dans le document parent. L'exemple ci-dessous montre comment un script qui s'exécute dans un document enfant peut présenter un objet contenant une fonction ou une propriété à son parent :

```
 var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Si cet enfant a été chargé dans une iframe avec une id "child", vous pourriez accéder à l'interface à partir du contenu parent en lisant la propriété `childSandboxBridge` de l'image :

```
 var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

Création d'un pont de sandbox parent

Adobe AIR 1.0 et les versions ultérieures

La propriété `parentSandboxBridge` permet au document parent de présenter une interface au contenu dans le document enfant. Pour présenter une interface, le document parent définit la propriété `parentSandbox` du document enfant sur une fonction ou un objet spécifié dans le document parent. Vous pouvez alors accéder à l'objet ou à la fonction à partir du contenu dans le document enfant. L'exemple ci-dessous montre comment un script qui s'exécute dans une image parent peut présenter à un document enfant un objet contenant une fonction :

```
 var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

A l'aide de cette interface, un contenu dans l'image enfant pourrait enregistrer du texte dans un fichier appelé `save.txt` mais il n'aurait aucun autre accès au système de fichiers. Le contenu enfant pourrait appeler la fonction save comme suit :

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Le contenu de l'application devrait présenter l'interface la plus étroite possible aux autres sandbox. Un contenu hors application devrait être considéré comme douteux par nature étant donné qu'il peut être sujet à une introduction de code accidentel ou malveillant. Vous devez mettre en place des mesures de protection appropriées pour éviter un usage abusif de l'interface que vous présentez par le pont de sandbox parent.

Accès à un pont de sandbox parent lors du chargement d'une page

Adobe AIR 1.0 et les versions ultérieures

Pour qu'un script dans un document enfant accède à un pont de sandbox parent, le pont doit être paramétré avant l'exécution du script. Les objets fenêtre, image et iframe distribuent un événement `dominitialize` lorsqu'une page DOM est créée, mais avant qu'un script ne soit analysé ou des éléments DOM ajoutés. Vous pouvez utiliser l'événement `dominitialize` pour établir le pont suffisamment tôt dans la séquence de création de la page afin que tous les scripts du document enfant puissent y accéder.

L'exemple ci-dessous montre comment créer un pont de sandbox parent en réponse à l'événement `dominitialize` distribué à partir de l'image enfant :

```
<html>
<head>
<script>
var bridgeInterface = {};
bridgeInterface.testProperty = "Bridge engaged";
function engageBridge(){
    document.getElementById("sandbox").contentWindow.parentSandboxBridge = bridgeInterface;
}
</script>
</head>
<body>
<iframe id="sandbox"
            src="http://www.example.com/air/child.html"
            documentRoot="app:/"
            sandboxRoot="http://www.example.com/air/"
            ondominitialize="engageBridge()"/>
</body>
</html>
```

Le document `child.html` suivant montre comment un contenu enfant peut accéder au pont de sandbox parent :

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Pour écouter l'événement `dominitialize` sur une fenêtre enfant plutôt qu'une image, il vous faut ajouter l'écouteur à l'objet window enfant créé par la fonction`window.open()` :

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

Il n'y a alors pas moyen de mapper le contenu de l'application dans un sandbox hors application. Cette technique n'est utile que lors du chargement de `child.html` hors du répertoire de l'application. Vous pouvez néanmoins mapper un contenu de l'application dans un sandbox hors application, mais il vous faut d'abord charger une page intermédiaire, qui elle-même utilise des images, afin de charger le document enfant et le mapper dans le sandbox souhaité.

Si vous utilisez la fonction `createRootWindow()` de la classe HTMLLoader pour créer une fenêtre, celle-ci n'est pas un enfant du document à partir duquel `createRootWindow()` est appelée. Ainsi, il n'est pas possible de créer un pont de sandbox à partir de la fenêtre appelante dans un contenu hors application chargé dans la nouvelle fenêtre. Il faut plutôt charger une page intermédiaire dans la nouvelle fenêtre qui elle-même utilise des images pour charger le document enfant. Vous pouvez alors établir le pont à partir du document parent de la nouvelle fenêtre avec le document enfant chargé dans l'image.

Chapitre 60 : Programmation du conteneur HTML d'AIR à l'aide de scripts

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader sert de conteneur au contenu HTML dans Adobe® AIR®. La classe fournit de nombreuses méthodes et propriétés, héritées de la classe Sprite, pour contrôler le comportement et l'aspect de l'objet dans la liste d'affichage d'ActionScript® 3.0. De plus, la classe définit les méthodes et propriétés pour les tâches telles que le chargement et l'interaction avec le contenu HTML et la gestion de l'historique.

La classe HTMLHost définit un ensemble de comportements associés par défaut à un objet HTMLLoader. Lorsque vous créez un objet HTMLLoader, aucune mise en œuvre de HTMLHost n'est fournie. Ainsi, lorsque le contenu HTML déclenche l'un des comportements par défaut, comme un changement d'emplacement de la fenêtre ou du titre de la fenêtre, rien ne se passe. Vous pouvez étendre la classe HTMLHost pour définir les comportements souhaités pour votre application.

Une mise en œuvre par défaut du HTMLHost est fournie pour les fenêtres créées par AIR. Vous pouvez affecter la mise en œuvre de HTMLHost par défaut à un autre objet HTMLLoader en paramétrant la propriété `htmlHost` de l'objet à l'aide d'un nouvel objet HTMLHost créé avec le paramètre `defaultBehavior` défini sur `true`.

Remarque : *dans Adobe® Flex™ Framework, l'objet HTMLLoader est encadré par le composant mx:HTML. Lorsque vous utilisez Flex, faites appel au composant HTML.*

Affichage des propriétés des objets HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

Un objet HTMLLoader hérite des propriétés d'affichage de la classe Adobe® Flash® Player Sprite. Vous pouvez redimensionner, déplacer, masquer et changer la couleur d'arrière-plan, par exemple. Ou vous pouvez appliquer des effets avancés comme les filtres, les masques, les mises à l'échelle et la rotation. Lorsque vous appliquez des effets, évaluez l'impact sur la lisibilité. Les contenus SWF et PDF chargés dans une page HTML ne peuvent pas s'afficher lorsque certains effets sont appliqués.

Les fenêtres HTML contiennent un objet HTMLLoader qui restitue le contenu HTML. L'objet est limité à la zone de la fenêtre de sorte qu'un changement dans les dimensions, la position, la rotation ou l'échelle ne produit pas toujours les résultats escomptés.

Propriétés d'affichage de base

Adobe AIR 1.0 et les versions ultérieures

Les propriétés d'affichage de base de HTMLLoader vous permettent de positionner le contrôle au sein de l'objet d'affichage parent, de définir la taille et d'afficher ou masquer le contrôle. Vous ne devriez pas changer ces objets pour l'objet HTMLLoader d'une fenêtre HTML.

Les propriétés de base contiennent :

Propriété	Remarques
`x`, `y`	Positionne l'objet au sein de son conteneur parent.
`width`, `height`	Change les dimensions de la zone d'affichage.
`visible`	Contrôle la visibilité de l'objet et tout contenu qu'il possède.

Hors d'une fenêtre HTML, les propriétés `width` et `height` d'un objet HTMLLoader sont fixées à 0. Vous devez définir la largeur et la hauteur avant que le contenu HTML chargé ne devienne visible. Le contenu HTML est dessiné suivant la taille de HTMLLoader et disposé suivant les propriétés HTML et CSS du contenu. Un changement dans la taille de HTMLLoader redistribue le contenu.

Lors du chargement de contenu dans un nouvel objet HTMLLoader (la propriété `width` est toujours définie sur 0), il peut être tentant de définir les paramètres `width` et `height` du HTMLLoader à l'aide des propriétés `contentWidth` et `contentHeight`. Cette technique est valable pour les pages qui disposent d'une largeur minimale raisonnable lorsqu'elles sont déployées selon les règles de HTML et du flux CSS. Toutefois, en l'absence d'une largeur raisonnable fournie par le HTMLLoader, certaines pages sont intégrées en une mise en page longue et étroite.

Remarque : *lorsque vous changez la largeur et la hauteur de l'objet HTMLLoader, les valeurs scaleX et scaleY ne changent pas, comme ce serait le cas avec la plupart des autres objets d'affichage.*

Transparence du contenu HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

La propriété `paintsDefaultBackground` d'un objet HTMLLoader, qui est `true` par défaut, détermine si l'objet HTMLLoader dessine un arrière-plan opaque. Lorsque `paintsDefaultBackground` est `false`, l'arrière-plan est clair. Le conteneur de l'objet d'affichage ou les autres objets d'affichage en dessous de l'objet HTMLLoader sont visibles derrière les éléments de premier plan du contenu HTML.

Si l'élément de corps ou tout autre élément du document HTML spécifie une couleur d'arrière-plan, à l'aide de `style="background-color:gray"`, par exemple, l'arrière-plan de cette portion de HTML est alors opaque et restituée dans la couleur d'arrière-plan spécifiée. Si vous définissez la propriété `opaqueBackground` de l'objet HTMLLoader et `paintsDefaultBackground` est `false`, la couleur définie pour le `opaqueBackground` est alors visible.

Remarque : *vous pouvez utiliser une illustration transparente et au format PNG pour fournir un arrière-plan semi-transparent à un élément dans un document HTML. La définition du style d'opacité d'un élément HTML n'est pas prise en charge.*

Mise à l'échelle d'un contenu HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

Evitez la mise à l'échelle d'un objet HTMLLoader au delà d'un facteur 1.0. Le texte d'un contenu HTMLLoader est restitué à une résolution spécifique et il apparaît pixelisé si l'échelle de l'objet HTMLLoader est augmentée. Pour éviter que le HTMLLoader, ainsi que son contenu, ne soit redimensionné lorsqu'une fenêtre l'est, définissez la propriété `scaleMode` de la scène sur `StageScaleMode.NO_SCALE`.

Eléments à prendre en compte lors du chargement d'un contenu SWF ou PDF dans une page HTML

Adobe AIR 1.0 et les versions ultérieures

Le contenu SWF ou PDF chargé dans un objet HTMLLoader disparaît dans les conditions suivantes :

- Si vous mettez l'objet HTMLLoader à une échelle autre que 1.0.
- Si vous attribuez à la propriété alpha de l'objet HTMLLoader une valeur autre que 1.0.
- Si vous faites pivoter le contenu HTMLLoader.

Le contenu réapparaît si vous retirez le paramétrage de la propriété fautive et supprimez les filtres actifs.

Par ailleurs, le moteur d'exécution ne peut pas afficher de contenu PDF dans une fenêtre transparente. Il n'affiche le contenu SWF intégré à une page HTML que si le paramètre `wmode` de l'objet ou de la balise embed est défini sur `opaque` ou `transparent`. Puisque la valeur par défaut de `wmode` est `window`, le contenu SWF ne s'affiche pas dans une fenêtre transparente, sauf si vous définissez explicitement le paramètre wmode.

Remarque : *dans les versions d'AIR antérieures à 1.5.2, le contenu SWF intégré à HTML ne s'affichait pas, quelle que soit la valeur wmode définie.*

Pour plus d'informations sur le chargement de ces types de média dans un objet HTMLLoader, voir les sections « Intégration d'un contenu SWF en HTML » à la page 1031 et « Ajout d'un contenu PDF dans AIR » à la page 569.

Propriétés d'affichage avancées

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader hérite de plusieurs méthodes qui peuvent être utilisées pour des effets spéciaux. En règle générale, ces effets ont des limites lorsqu'ils sont utilisés avec l'affichage du HTMLLoader, mais ils peuvent être utiles pour les transitions ou autres effets temporaires. Par exemple, si vous affichez une fenêtre de dialogue pour rassembler les saisies de l'utilisateur, vous pourriez flouter l'affichage de la fenêtre principale jusqu'à ce que celui-ci mette fin au dialogue. De même, vous pourriez faire un fondu lors de la fermeture de la fenêtre

Les propriétés d'affichage avancées sont les suivantes :

Propriété	Restrictions
`alpha`	Peut réduire la lisibilité du contenu HTML
`filtres`	Dans une fenêtre HTML, les effets extérieurs sont coupés par les bords de la fenêtre
`graphics`	Les formes dessinées à l'aide de commandes graphiques apparaissent en dessous du contenu HTML, ainsi que l'arrière-plan par défaut. La propriété paintsDefaultBackground doit être « false » pour que les formes dessinées soient visibles.
`opaqueBackground`	Ne change pas la couleur de l'arrière-plan par défaut. La propriété paintsDefaultBackground doit être "false" pour que cette couche de couleur soit visible.

Propriété	Restrictions
`rotation`	Les coins de la zone HTMLLoader rectangulaire peuvent être découpées par le bord de fenêtre. Le contenu SWF ou PDF chargé dans le contenu HTML n'est pas affiché.
`scaleX`, `scaleY`	L'affichage restitué peut apparaître pixelisé pour des échelles supérieures à 1. Le contenu SWF ou PDF chargé dans le contenu HTML n'est pas affiché.
`transform`	Peut réduire la lisibilité du contenu HTML L'affichage HTML peut être coupé par le bord de la fenêtre. Le contenu SWF ou PDF chargé dans le contenu HTML n'est pas affiché si la transformation implique un pivotement, un redimensionnement ou une inclinaison.

L'exemple suivant illustre la façon de paramétrer le tableau `filters` pour flouter l'affichage HTML au complet :

```
 var html:HTMLLoader = new HTMLLoader();
var urlReq:URLRequest = new URLRequest("http://www.adobe.com/");
html.load(urlReq);
html.width = 800;
html.height = 600;

var blur:BlurFilter = new BlurFilter(8);
var filters:Array = [blur];
html.filters = filters;
```

Défilement du contenu HTML

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader contient les propriétés suivantes qui vous permettent de contrôler le défilement du contenu HTML :

Propriété	Description
`contentHeight`	La hauteur, exprimée en pixels, du contenu HTML.
contentWidth	La largeur, exprimée en pixels, du contenu HTML.
scrollH	La position de défilement horizontal du contenu HTML au sein de l'objet HTMLLoader.
scrollV	La position de défilement vertical du contenu HTML au sein de l'objet HTMLLoader.

Le code ci-dessous définit la propriété `scrollV` pour que vous puissiez faire défiler le contenu HTML jusqu'au bas de la page :

```
 var html:HTMLLoader = new HTMLLoader();
html.addEventListener(Event.HTML_BOUNDS_CHANGE, scrollHTML);

const SIZE:Number = 600;
html.width = SIZE;
html.height = SIZE;

var urlReq:URLRequest = new URLRequest("http://www.adobe.com");
html.load(urlReq);
this.addChild(html);

function scrollHTML(event:Event):void
{
    html.scrollV = html.contentHeight - SIZE;
}
```

Le HTMLLoader ne contient pas les barres de défilement horizontal et vertical. Vous pouvez implémenter des barres de défilement dans ActionScript ou à l'aide d'un composant Flex. Le composant HTML Flex contient automatiquement des barres de défilement pour du contenu HTML. Vous pouvez également utiliser la méthode `HTMLLoader.createRootWindow()` pour créer une fenêtre qui contienne un objet HTMLLoader avec des barres de défilement. Voir la section « Création de fenêtres avec défilement de contenu HTML » à la page 1059.

Accès à l'historique de HTML

Adobe AIR 1.0 et les versions ultérieures

A mesure que de nouvelles pages sont chargées dans un objet HTMLLoader, le moteur d'exécution actualise un historique pour l'objet. L'historique correspond à l'objet `window.history` dans la page HTML. La classe HTMLLoader contient les méthodes et propriétés suivantes qui vous permettent d'utiliser l'historique de HTML :

Membre de la classe	Description
`historyLength`	La longueur hors tout de l'historique englobant les entrées précédentes et suivantes.
`historyPosition`	La position actuelle dans l'historique. Les éléments de l'historique qui se trouvent avant cette position correspondent à la navigation « précédente » et ceux qui se trouvent après, à la navigation « suivante ».
`getHistoryAt()`	Renvoie l'objet URLRequest correspondant à l'entrée qui se trouve à la position spécifiée de l'historique.
`historyBack()`	Navigue en amont dans l'historique, si possible.
`historyForward()`	Navigue en aval dans l'historique, si possible.
`historyGo()`	Se déplace dans l'historique selon le nombre d'étapes indiqué. Navigue en aval dans l'historique si la valeur est positive ou en amont dans l'autre cas. Une valeur zéro conduit à un rechargement de la page. La spécification d'une position qui se trouve au-delà de la fin conduit à la dernière entrée de la liste.

Les éléments figurant dans l'historique sont stockés en tant qu'objets de type HTMLHistoryItem. La classe HTMLHistoryItem possède les propriétés suivantes :

Propriété	Description
`isPost`	Définissez sur `true` si la page HTML contient des données POST.
`originalUrl`	L'URL d'origine de la page HTML, avant toute redirection.
`title`	Le titre de la page HTML.
`url`	L'URL de la page HTML.

Paramétrage de l'agent d'utilisateur employé lors du chargement du contenu HTML

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader possède une propriété `userAgent` qui vous permet de définir la chaîne de l'agent d'utilisateur employé par HTMLLoader. Paramétrez la propriété `userAgent` de l'objet HTMLLoader avant d'appeler la méthode `load()`. Si vous définissez cette propriété sur l'occurrence de HTMLLoader, la propriété `userAgent` de l'URLRequest transmise à la méthode `load()` n'est alors *pas* utilisée.

Vous pouvez définir la chaîne de l'agent d'utilisateur employée par tous les objets HTMLLoader dans un domaine d'application en paramétrant la propriété `URLRequestDefaults.userAgent`. Les propriétés URLRequestDefaults statiques sont appliquées par défaut à tous les objets URLRequest, et pas seulement aux objets URLRequest utilisés avec la méthode `load()` des objets HTMLLoader. Le paramétrage de la propriété `userAgent` d'un HTMLLoader remplace celui par défaut de `URLRequestDefaults.userAgent`.

Si vous ne paramétrez pas une valeur de l'agent d'utilisateur pour la propriété `userAgent` de l'objet HTMLLoader ou pour `URLRequestDefaults.userAgent`, la valeur de l'agent d'utilisateur AIR par défaut est employée. Cette valeur par défaut varie suivant le système d'exploitation utilisé à l'exécution (Mac OS ou Windows, par exemple), le langage du moteur d'exécution et sa version, comme dans les deux exemples suivants :

- `"Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) AdobeAIR/1.0"`
- `"Mozilla/5.0 (Windows; U; en) AppleWebKit/420+ (KHTML, like Gecko) AdobeAIR/1.0"`

Paramétrage du codage des caractères à utiliser pour le contenu HTML

Adobe AIR 1.0 et les versions ultérieures

Une page HTML peut spécifier le codage des caractères qu'elle utilise en incluant la balise `meta` comme suit :

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

Remplacez le paramétrage de la page pour vous assurer que le codage d'un caractère spécifique est utilisé en définissant la propriété `textEncodingOverride` de l'objet HTMLLoader :

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

Spécifiez le codage de caractère pour le contenu HTMLLoader à utiliser lorsqu'une page HTML ne spécifie pas un paramétrage avec la propriété `textEncodingFallback` de l'objet HTMLLoader :

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

La propriété `textEncodingOverride` remplace le paramétrage de la page HTML. Et la propriété `textEncodingOverride` ainsi que le paramétrage de la page HTML remplacent la propriété `textEncodingFallback`.

Paramétrez l'une ou l'autre de ces propriétés avant le chargement du contenu HTML.

Paramétrage des interfaces utilisateur de type navigateur pour un contenu HTML

Adobe AIR 1.0 et les versions ultérieures

JavaScript propose plusieurs interfaces de programmation pour contrôler la fenêtre qui affiche le contenu HTML. Dans AIR, vous pouvez annuler ces API en implémentant une classe HTMLHost personnalisée.

A propos de l'extension de la classe HTMLHost

Adobe AIR 1.0 et les versions ultérieures

Si, par exemple, votre application présente plusieurs objets HTMLLoader dans une interface à onglets, vous pourriez souhaiter changer le libellé d'un onglet à l'aide des pages HTML chargées, sans toucher au titre de la fenêtre principale. De même, votre code pourrait répondre à un appel `window.moveTo()` en repositionnant l'objet HTMLLoader dans son conteneur d'objet d'affichage parent, en déplaçant la fenêtre qui contient l'objet HTMLLoader, en ne faisant rien du tout ou en faisant tout autre chose.

La classe HTMLHost d'AIR contrôle les méthodes et propriétés JavaScript suivantes :

- `window.status`
- `window.document.title`
- `window.location`
- `window.blur()`
- `window.close()`
- `window.focus()`
- `window.moveBy()`
- `window.moveTo()`
- `window.open()`
- `window.resizeBy()`
- `window.resizeTo()`

Lorsque vous créez un objet HTMLLoader à l'aide de `new HTMLLoader()`, les propriétés ou méthodes répertoriées ne sont pas activées. La classe HTMLHost fournit par défaut une implémentation de ces interfaces de programmation JavaScript, similaire à celle des navigateurs. Vous pouvez aussi étendre la classe HTMLHost pour personnaliser le comportement. Pour créer un objet HTMLHost qui prenne en charge le comportement par défaut, définissez le paramètre `defaultBehaviors` sur true dans le constructeur HTMLHost.

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

Lorsque vous créez une fenêtre HTML dans AIR à l'aide de la méthode `createRootWindow()` de la classe HTMLLoader, une occurrence de HTMLHost qui prend en charge les comportements est automatiquement affectée. Vous pouvez changer le comportement de l'objet hôte en affectant une implémentation HTMLLoader différente à la propriété `htmlHost` du HTMLLoader ou bien vous pouvez affecter `null` pour désactiver complètement les fonctions.

Remarque : *AIR affecte un objet HTMLHost par défaut à la fenêtre initiale créée pour une application AIR à base d'HTML et à toute fenêtre créée par l'implémentation par défaut de la méthode `window.open()` JavaScript.*

Exemple : Extension de la classe HTMLHost

Adobe AIR 1.0 et les versions ultérieures

L'exemple ci-dessous montre comment personnaliser l'affectation de l'interface utilisateur par un objet HTMLLoader en étendant la classe HTMLLoader :

Exemple Flex :

1. Créez une classe qui étend la classe HTMLHost (une sous-classe).
2. Remplacez les méthodes de la nouvelle classe pour gérer les changements dans les paramètres de l'utilisateur liés à l'interface. Par exemple, la classe suivante, CustomHost, définit les comportements pour les appels à`window.open()` et les changements à `window.document.title`. Les appels à `window.open()` ouvrent la page HTML dans une nouvelle fenêtre et les changements à `window.document.title`, y compris le paramétrage de l'élément `<title>` d'une page HTML, définissent le titre de cette fenêtre.

```
package
{
    import flash.html.*;
    import flash.display.StageScaleMode;
    import flash.display.NativeWindow;
    import flash.display.NativeWindowInitOptions;

    public class CustomHost extends HTMLHost
    {
        import flash.html.*;
        override public function
            createWindow(windowCreateOptions:HTMLWindowCreateOptions):HTMLLoader
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(windowCreateOptions.x,
                                                 windowCreateOptions.y,
                                                 windowCreateOptions.width,
                                                 windowCreateOptions.height);
            var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                                windowCreateOptions.scrollBarsVisible, bounds);
            htmlControl.htmlHost = new HTMLHostImplementation();
            if(windowCreateOptions.fullscreen){
                htmlControl.stage.displayState =
                    StageDisplayState.FULL_SCREEN_INTERACTIVE;
            }
            return htmlControl;
        }
        override public function updateTitle(title:String):void
        {
            htmlLoader.stage.nativeWindow.title = title;
        }
    }
}
```

3 Dans le code qui contient le HTMLLoader (pas le code de la nouvelle sous-classe de HTMLHost), créez un objet de la nouvelle classe. Affectez le nouvel objet à la propriété `htmlHost` du HTMLLoader. Le code Flex ci-dessous utilise la classe CustomHost définie à l'étape précédente :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication
    xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    applicationComplete="init()">
    <mx:Script>
        <![CDATA[
            import flash.html.HTMLLoader;
            import CustomHost;
            private function init():void
            {
                var html:HTMLLoader = new HTMLLoader();
                html.width = container.width;
                html.height = container.height;
                var urlReq:URLRequest = new URLRequest("Test.html");
                html.htmlHost = new CustomHost();
                html.load(urlReq);
                container.addChild(html);
            }
        ]]>
    </mx:Script>
    <mx:UIComponent id="container" width="100%" height="100%"/>
</mx:WindowedApplication>
```

Pour tester le code décrit ici, incluez un fichier HTML qui ait le contenu suivant dans le répertoire de l'application :

```
<html>
    <head>
        <title>Test</title>
    </head>
    <script>
        function openWindow()
        {
            window.runtime.trace("in");
            document.title = "foo"
            window.open('Test.html');
            window.runtime.trace("out");
        }
    </script>
    <body>
        <a href="#" onclick="openWindow()">window.open('Test.html')</a>
    </body>
</html>
```

Exemple Flash Professional :

1. Créez un fichier Flash pour AIR. Définissez sa classe de document sur CustomHostExample, puis enregistrez le fichier sous CustomHostExample.fla.
2. Créez un fichier ActionScript appelé CustomHost.as contenant une classe qui étend la classe HTMLHost (une sous-classe). Cette classe remplace certaines méthodes de la nouvelle classe pour traiter les changements dans les paramètres de l'utilisateur liés à l'interface. Par exemple, la classe suivante, CustomHost, définit les comportements pour les appels à`window.open()` et les changements à `window.document.title`. Les appels à la méthode `window.open()` ouvrent la page HTML dans une nouvelle fenêtre et les changements à la propriété `window.document.title`, y compris le paramétrage de l'élément `<title>` d'une page HTML, définissent le titre de cette fenêtre.

```
package
{
    import flash.display.StageScaleMode;
    import flash.display.NativeWindow;
    import flash.display.NativeWindowInitOptions;
    import flash.events.Event;
    import flash.events.NativeWindowBoundsEvent;
    import flash.geom.Rectangle;
    import flash.html.HTMLLoader;
    import flash.html.HTMLHost;
    import flash.html.HTMLWindowCreateOptions;
    import flash.text.TextField;

    public class CustomHost extends HTMLHost
    {
        public var statusField:TextField;

        public function CustomHost(defaultBehaviors:Boolean=true)
        {
            super(defaultBehaviors);
        }
        override public function windowClose():void
        {
            htmlLoader.stage.nativeWindow.close();
        }
        override public function createWindow(
                                windowCreateOptions:HTMLWindowCreateOptions ):HTMLLoader
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(windowCreateOptions.x,
                                            windowCreateOptions.y,
                                            windowCreateOptions.width,
                                            windowCreateOptions.height);
            var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                        windowCreateOptions.scrollBarsVisible, bounds);
            htmlControl.htmlHost = new HTMLHostImplementation();
            if(windowCreateOptions.fullscreen){
                htmlControl.stage.displayState =
                    StageDisplayState.FULL_SCREEN_INTERACTIVE;
            }
            return htmlControl;
        }
        override public function updateLocation(locationURL:String):void
        {
            trace(locationURL);
        }
        override public function set windowRect(value:Rectangle):void
        {
            htmlLoader.stage.nativeWindow.bounds = value;
```

```
        }
        override public function updateStatus(status:String):void
        {
            statusField.text = status;
            trace(status);
        }
        override public function updateTitle(title:String):void
        {
            htmlLoader.stage.nativeWindow.title = title + "- Example Application";
        }
        override public function windowBlur():void
        {
            htmlLoader.alpha = 0.5;
        }
        override public function windowFocus():void
        {
            htmlLoader.alpha = 1;
        }
    }
}
```

3 Créez un autre fichier ActionScript appelé CustomHostExample.as pour contenir la classe document pour cette application. Cette classe crée un objet HTMLLoader et définit sa propriété hôte sur une occurrence de la classe CustomHost définie à l'étape précédente.

```
package
{
    import flash.display.Sprite;
    import flash.html.HTMLLoader;
    import flash.net.URLRequest;
    import flash.text.TextField;

    public class CustomHostExample extends Sprite
    {
        function CustomHostExample():void
        {
            var html:HTMLLoader = new HTMLLoader();
            html.width = 550;
            html.height = 380;
            var host:CustomHost = new CustomHost();
            html.htmlHost = host;

            var urlReq:URLRequest = new URLRequest("Test.html");
            html.load(urlReq);

            addChild(html);

            var statusTxt:TextField = new TextField();
            statusTxt.y = 380;
            statusTxt.height = 20;
            statusTxt.width = 550;
            statusTxt.background = true;
            statusTxt.backgroundColor = 0xEEEEEEEE;
            addChild(statusTxt);

            host.statusField = statusTxt;
        }
    }
}
```

Pour tester le code décrit ici, incluez un fichier HTML qui ait le contenu suivant dans le répertoire de l'application :

```
<html>
     <head>
     <title>Test</title>
     <script>
     function openWindow()
     {
     document.title = "Test"
     window.open('Test.html');
     }
     </script>
     </head>
     <body bgColor="#EEEEEE">
     <a href="#" onclick="window.open('Test.html')">window.open('Test.html')</a>
     <br/><a href="#" onclick="window.document.location='http://www.adobe.com'">
     window.document.location = 'http://www.adobe.com'</a>
     <br/><a href="#" onclick="window.moveBy(6, 12)">moveBy(6, 12)</a>
     <br/><a href="#" onclick="window.close()">window.close()</a>
     <br/><a href="#" onclick="window.blur()">window.blur()</a>
     <br/><a href="#" onclick="window.focus()">window.focus()</a>
     <br/><a href="#" onclick="window.status = new Date().toString()">window.status=new
Date().toString()</a>
     </body>
</html>
```

Traitement des changements apportés à la propriété window.location

Adobe AIR 1.0 et les versions ultérieures

Remplacez la méthode `locationChange()` pour traiter les changements apportés à l'URL de la page HTML. La méthode `locationChange()` est appelée lorsque JavaScript dans une page change la valeur de `window.location`. L'exemple ci-dessous charge tout simplement l'URL demandée :

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

Remarque : *vous pouvez utiliser la propriété HTMLLoader de l'objet HTMLHost pour faire référence à l'objet HTMLLoader actuel.*

Traitement des appels JavaScript à window.moveBy(), window.moveTo(), window.resizeTo(), window.resizeBy()

Adobe AIR 1.0 et les versions ultérieures

Remplacez la méthode `set windowRect()` pour traiter les changements dans les limites du contenu HTML. La méthode `set windowRect()` est appelée lorsque JavaScript dans une page appelle `window.moveBy()`, `window.moveTo()`, `window.resizeTo()` ou `window.resizeBy()`. L'exemple ci-dessous met simplement à jour les limites de la fenêtre de l'ordinateur de bureau :

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

Traitement des appels JavaScript à window.open()

Adobe AIR 1.0 et les versions ultérieures

Remplacez la méthode `createWindow()` pour traiter les appels JavaScript à `window.open()`. Les implémentations de la méthode `createWindow()` sont responsables de la création d'un objet HTMLLoader et de son renvoi. Vous afficheriez généralement le HTMLLoader dans une nouvelle fenêtre, mais la création d'une fenêtre n'est pas nécessaire.

L'exemple suivant décrit comment implémenter la fonction `createWindow()` à l'aide du `HTMLLoader.createRootWindow()` pour créer à la fois la fenêtre et l'objet HTMLLoader. Vous pouvez aussi créer un objet NativeWindow à part et ajouter le HTMLLoader à la scène de la fenêtre.

```
override public function createWindow(windowCreateOptions:HTMLWindowCreateOptions):HTMLLoader{
    var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
    var bounds:Rectangle = new Rectangle(windowCreateOptions.x, windowCreateOptions.y,
                                windowCreateOptions.width, windowCreateOptions.height);
    var htmlControl:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions,
                                    windowCreateOptions.scrollBarsVisible, bounds);
    htmlControl.htmlHost = new HTMLHostImplementation();
    if(windowCreateOptions.fullscreen){
        htmlControl.stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
    }
    return htmlControl;
}
```

Remarque : *cet exemple affecte l'implémentation personnalisée de HTMLHost à toute fenêtre créée avec `window.open()`. Vous pouvez également utiliser une implémentation différente ou définir la propriété `htmlHost` sur "null" pour de nouvelles fenêtres, si vous le souhaitez.*

L'objet transmis en tant que paramètre à la méthode `createWindow()` est un objet HTMLWindowCreateOptions. La classe HTMLWindowCreateOptions inclut des propriétés qui signalent les valeurs définies dans la chaîne du paramètre `features`, dans l'appel à `window.open()` :

Propriété HTMLWindowCreateOptions	Paramètre correspondant dans la chaîne des fonctions se trouvant dans l'appel JavaScript à window.open()
`fullscreen`	`fullscreen`
`height`	`height`
`locationBarVisible`	`location`
`menuBarVisible`	`menubar`
`resizeable`	`resizable`
`scrollBarsVisible`	`scrollbars`
`statusBarVisible`	`status`
`toolBarVisible`	`toolbar`
`width`	`width`
`x`	`left` ou `screenX`
`y`	`top` ou `screenY`

La classe HTMLLoader n'implémente pas toutes les fonctions susceptibles d'être spécifiées dans la chaîne de fonctions. Votre application doit fournir des barres de défilement, des barres d'adresses, des barres de menus, des barres d'état et des barres d'outils lorsque cela convient.

Les autres arguments pour la méthode `window.open()` de JavaScript sont traités par le système. Une implémentation `createWindow()` ne devrait pas charger de contenu dans l'objet HTMLLoader ou définir le titre de la fenêtre.

Traitement des appels JavaScript à window.close()

Adobe AIR 1.0 et les versions ultérieures

Remplacez `window.close()` pour que les appels JavaScript soient traités par la méthode `window.close()`. L'exemple ci-dessous ferme la fenêtre de l'ordinateur de bureau lorsque la méthode `window.close()` est appelée :

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

Les appels JavaScript à `window.close()` n'ont pas besoin de fermer la fenêtre de contenu. Vous pourriez, par exemple, retirer HTMLLoader de la liste d'affichage en laissant la fenêtre (qui peut avoir un autre contenu) ouverte, comme dans le code suivant :

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

Traitement des changements dans la propriété de l'état de la fenêtre

Adobe AIR 1.0 et les versions ultérieures

Remplacez la méthode `updateStatus()` pour traiter les changements JavaScript à la valeur de `window.status`. L'exemple ci-dessous assure le suivi de la valeur d'état :

```
override public function updateStatus(status:String):void
{
    trace(status);
}
```

L'état demandé est transmis en tant que chaîne à la méthode `updateStatus()`.

L'objet HTMLLoader ne fournit pas de barre d'état.

Traitement des changements dans la propriété window.document.title

Adobe AIR 1.0 et les versions ultérieures

Remplacez la méthode `updateTitle()` pour traiter les changements JavaScript à la valeur de `window.document.title`. L'exemple ci-dessous change le titre de la fenêtre et ajoute la chaîne "Sample" au titre :

```
override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

Lorsque `document.title` est défini sur une page HTML, le titre demandé est transmis en tant que chaîne à la méthode `updateTitle()`.

Il n'est pas nécessaire que les changements à `document.title`modifient le titre de la fenêtre contenant l'objet HTMLLoader. Vous pourriez, par exemple, changer un autre élément d'interface, tel qu'un champ de texte.

Traitement des appels JavaScript à window.blur() et window.focus()

Adobe AIR 1.0 et les versions ultérieures

Remplacez les méthodes `windowBlur()` et `windowFocus()` pour traiter les appels JavaScript à `window.blur()` et `window.focus()`, comme le montre l'exemple ci-dessous :

```
override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

Remarque : *AIR ne fournit pas d'interface de programmation pour désactiver une fenêtre ou une application.*

Création de fenêtres avec défilement de contenu HTML

Adobe AIR 1.0 et les versions ultérieures

La classe HTMLLoader inclut une méthode statique, `HTMLLoader.createRootWindow()`, vous permettant d'ouvrir une nouvelle fenêtre (représentée par un objet NativeWindow) qui contient un object HTMLLoader et qui définit quelques paramètres d'interface utilisateur pour cette fenêtre. Il faut à cette méthode quatre paramètres pour vous permettre de définir l'interface utilisateur.

Paramètre	Description
`visible`	Une valeur booléenne indiquant si la fenêtre est initialement visible (`true`) ou non (`false`).
`windowInitOptions`	Un objet NativeWindowInitOptions. La classe NativeWindowInitOptions définit les options d'initialisation pour un objet NativeWindow, y compris ce qui suit : si on peut réduire la fenêtre au minimum, la développer au maximum ou la redimensionner, si la fenêtre possède un chrome système ou un chrome personnalisable, si la fenêtre est transparente ou non (pour les fenêtres qui n'utilisent pas de chrome système) et le type de fenêtre.
`scrollBarsVisible`	Indique s'il existe des barres de défilement (`true`) ou non (`false`).
`bounds`	Un objet Rectangle qui définit la position et la taille de la nouvelle fenêtre.

Par exemple, le code ci-dessous fait appel à la méthode `HTMLLoader.createRootWindow()` pour créer une fenêtre dans laquelle figure le contenu HTMLLoader qui utilise des barres de défilement :

```
var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
var bounds:Rectangle = new Rectangle(10, 10, 600, 400);
var html2:HTMLLoader = HTMLLoader.createRootWindow(true, initOptions, true, bounds);
var urlReq2:URLRequest = new URLRequest("http://www.example.com");
html2.load(urlReq2);
html2.stage.nativeWindow.activate();
```

Remarque : *les fenêtres créées par un appel direct de* `createRootWindow()` *dans JavaScript demeurent indépendantes de la fenêtre HTML d'ouverture. Les propriétés JavaScript Window* `opener` *et* `parent`*, par exemple, sont* `null`*. Toutefois, si vous appelez* `createRootWindow()` *indirectement en remplaçant la méthode HTMLHost* `createWindow()` *pour appeler* `createRootWindow()`*,* `opener` *et* `parent` *font effectivement référence à la fenêtre HTML d'ouverture.*

Création de sous-classes de la classe HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez créer une sous-classe de la classe HTMLLoader pour créer des comportements. Par exemple, vous pouvez créer une sous-classe qui définisse des écouteurs d'événement par défaut pour les événements HTMLLoader, tels que ces événements distribués lorsque HTML est restitué ou que l'utilisateur clique sur un lien.

L'exemple suivant étend la classe HTMLHost pour fournir un comportement *normal* lorsque la méthode `window.open()` JavaScript est appelée. L'exemple définit alors une sous-classe de HTMLLoader qui utilise la classe d'implémentation du HTMLHost personnalisé :

```
 package
{
       import flash.html.HTMLLoader;
    public class MyHTMLHost extends HTMLHost
       {
           public function MyHTMLHost()
           {
               super(false);
        }
        override public function createWindow(opts:HTMLWindowCreateOptions):void
        {
            var initOptions:NativeWindowInitOptions = new NativeWindowInitOptions();
            var bounds:Rectangle = new Rectangle(opts.x, opts.y, opts.width, opts.height);
            var html:HTMLLoader = HTMLLoader.createRootWindow(true,
                                                  initOptions,
                                                  opts.scrollBarsVisible,
                                                  bounds);
            html.stage.nativeWindow.orderToFront();
            return html
       }
}
```

Le code ci-dessous définit une sous-classe de la classe HTMLLoader qui affecte un objet MyHTMLHost à sa propriété `htmlHost` :

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

Pour des détails sur la classe HTMLHost et la méthode `HTMLLoader.createRootWindow()` utilisée dans cet exemple, voir la section « Paramétrage des interfaces utilisateur de type navigateur pour un contenu HTML » à la page 1049.

Chapitre 61 : Gestion des événements HTML dans AIR

Adobe AIR 1.0 et les versions ultérieures

Un système de gestion des événements permet au programmeur de répondre aux actions de l'utilisateur et aux événements système de manière pratique. Le modèle d'événement Adobe® AIR® n'est pas seulement utile, il est également conforme aux standards en vigueur. Ce modèle repose sur la spécification d'événements Document Object Model (DOM) de niveau 3, une architecture de gestion d'événements normalisée. Il constitue donc pour les programmeurs un outil puissant, mais parfaitement intuitif.

Evénements HTMLLoader

Adobe AIR 1.0 et les versions ultérieures

Un objet HTMLLoader déclenche les événements Adobe® ActionScript® 3.0 suivants :

Evénement	Description
`htmlDOMInitialize`	Distribué lors de la création du document HTML, mais avant que tout script soit analysé ou que les nœuds DOM soient ajoutés à la page.
`complete`	Distribué lors de la création du DOM HTML en réponse à une opération de chargement, juste après l'événement `onload` sur la page HTML.
`htmlBoundsChanged`	Distribué lorsque l'une des propriétés `contentWidth` et `contentHeight` ou les deux à la fois ont été modifiées.
`locationChange`	Distribué lorsque la propriété location de l'objet HTMLLoader a été modifiée.
`locationChanging`	Distribué avant que l'emplacement de l'objet HTMLLoader ne change suite à la navigation de l'utilisateur, à un appel de JavaScript ou à une redirection. L'événement `locationChanging` n'est pas distribué lorsque vous appelez la méthode `load()`, `loadString()`, `reload()`, `historyGo()`, `historyForward()` ou `historyBack()`. L'appel de la méthode `preventDefault()` de l'objet d'événement distribué annule la navigation. Si un lien est ouvert dans le navigateur système, aucun événement locationChanging n'est distribué, car l'objet HTMLLoader ne change pas d'emplacement.
`scroll`	Distribué à chaque changement de position de défilement par le moteur HTML. Un événement scroll résulte d'un accès aux liens d'ancrage (liens #) sur la page ou d'un appel de la méthode `window.scrollTo()`. La saisie de texte dans une zone de texte ou d'entrée de texte peut également provoquer un événement scroll.
`uncaughtScriptException`	Distribué lorsqu'il se produit une exception JavaScript dans l'objet HTMLLoader et qu'elle n'est pas interceptée dans le code JavaScript.

Vous pouvez également associer une fonction ActionScript à un événement JavaScript (tel que `onClick`). Pour plus d'informations, voir la section « Gestion des événements DOM avec ActionScript » à la page 1063.

Gestion des événements DOM avec ActionScript

Adobe AIR 1.0 et les versions ultérieures

Vous pouvez enregistrer des fonctions ActionScript qui répondent à des événements JavaScript. Examinons par exemple le contenu HTML suivant :

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

Vous pouvez enregistrer une fonction ActionScript en tant que gestionnaire de tout événement de la page. Par exemple, le code suivant ajoute la fonction `clickHandler()` en tant qu'écouteur de l'événement `onclick` de l'élément `testLink` sur la page HTML :

```
var html:HTMLLoader = new HTMLLoader( );
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.addEventListener(Event.COMPLETE, completeHandler);

function completeHandler(event:Event):void {
    html.window.document.getElementById("testLink").onclick = clickHandler;
}

function clickHandler( event:Object ):void {
    trace("Event of type: " + event.type );
}
```

L'objet événement distribué n'est pas de type flash.events.Event et n'appartient pas à l'une des sous-classes Event. La classe Object permet de déclarer le type de l'argument de la fonction du gestionnaire d'événement.

Vous pouvez également utiliser la méthode `addEventListener()` pour enregistrer des fonctions associées à ces événements. Il est par exemple possible de remplacer la méthode `completeHandler()` dans l'exemple précédent par le code suivant :

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

Lorsqu'un écouteur se réfère à un élément DOM déterminé, il est recommandé d'attendre que l'objet HTMLLoader parent distribue l'événement `complete` avant d'ajouter les écouteurs d'événement. Les pages HTML chargent souvent plusieurs fichiers et le DOM HTML n'est pas totalement généré tant que tous les fichiers n'ont pas été chargés et analysés. Une fois tous les éléments créés, l'objet HTMLLoader distribue l'événement `complete`.

Réponse aux exceptions JavaScript non interceptées

Adobe AIR 1.0 et les versions ultérieures

Examinez le code HTML suivant :

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Il contient une fonction JavaScript, `throwError()`, qui fait référence à une variable inconnue, `melbaToast` :

```
var x = 400 * melbaToast;
```

Lorsqu'une opération JavaScript rencontre une opération non valide qui n'est pas interceptée dans le code JavaScript par une structure `try/catch`, l'objet HTMLLoader contenant la page distribue un événement HTMLUncaughtScriptExceptionEvent. Vous pouvez enregistrer un gestionnaire associé à cet événement, comme illustré par le code suivant :

```
var html:HTMLLoader = new HTMLLoader();
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.width = container.width;
html.height = container.height;
container.addChild(html);
html.addEventListener(HTMLUncaughtScriptExceptionEvent.UNCAUGHT_SCRIPT_EXCEPTION,
                          htmlErrorHandler);
function htmlErrorHandler(event:HTMLUncaughtJavaScriptExceptionEvent):void
{
    event.preventDefault();
    trace("exceptionValue:", event.exceptionValue)
    for (var i:int = 0; i < event.stackTrace.length; i++)
    {
        trace("sourceURL:", event.stackTrace[i].sourceURL);
        trace("line:", event.stackTrace[i].line);
        trace("function:", event.stackTrace[i].functionName);
    }
}
```

JavaScript permet de gérer le même événement à l'aide de la propriété window.htmlLoader :

```
<html>
<head>
<script language="javascript" type="text/javascript" src="AIRAliases.js"></script>

    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }

        function htmlErrorHandler(event) {
            event.preventDefault();
            var message = "exceptionValue:" + event.exceptionValue + "\n";
            for (var i = 0; i < event.stackTrace.length; i++){
                message += "sourceURL:" + event.stackTrace[i].sourceURL +"\n";
                message += "line:" + event.stackTrace[i].line +"\n";
                message += "function:" + event.stackTrace[i].functionName + "\n";
            }
            alert(message);
        }

        window.htmlLoader.addEventListener("uncaughtScriptException", htmlErrorHandler);
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Le gestionnaire d'événement `htmlErrorHandler()` annule le comportement par défaut de l'événement (qui consiste à envoyer le message d'erreur JavaScript à la sortie de suivi AIR) et génère son propre message de sortie. Il stipule la valeur `exceptionValue` de l'objet HTMLUncaughtScriptExceptionEvent. Il indique par ailleurs les propriétés de chaque objet dans le tableau `stackTrace` :

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

Gestion des événements d'exécution avec JavaScript

Adobe AIR 1.0 et les versions ultérieures

Les classes d'exécution gèrent l'ajout de gestionnaires d'événement à l'aide de la méthode `addEventListener()`. Pour ajouter une fonction de gestionnaire associée à un événement, appelez la méthode `addEventListener()` de l'objet qui distribue l'événement en indiquant le type d'événement et la fonction concernée. Par exemple, pour écouter l'événement `closing` distribué lorsqu'un utilisateur clique sur le bouton de fermeture de fenêtre dans la barre de titre, utilisez l'instruction suivante :

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

Création d'une fonction de gestionnaire d'événement

Adobe AIR 1.0 et les versions ultérieures

Le code suivant crée un fichier HTML simple qui affiche des informations sur la position de la fenêtre principale. Une fonction de gestionnaire appelée `moveHandler()` écoute un événement move (défini par la classe NativeWindowBoundsEvent) de la fenêtre principale.

```
<html>
    <script src="AIRAliases.js" />
    <script>
        function init() {
            writeValues();
            window.nativeWindow.addEventListener(air.NativeWindowBoundsEvent.MOVE,
                                                 moveHandler);
        }
        function writeValues() {
            document.getElementById("xText").value = window.nativeWindow.x;
            document.getElementById("yText").value = window.nativeWindow.y;
        }
        function moveHandler(event) {
            air.trace(event.type); // move
            writeValues();
        }
    </script>
    <body onload="init()" />
        <table>
            <tr>
                <td>Window X:</td>
                <td><textarea id="xText"></textarea></td>
            </tr>
            <tr>
                <td>Window Y:</td>
                <td><textarea id="yText"></textarea></td>
            </tr>
        </table>
    </body>
</html>
```

Lorsqu'un utilisateur déplace la fenêtre, les éléments textarea affichent les positions X et Y actualisées de la fenêtre.

Notez que l'objet événement est transmis sous forme d'argument à la méthode `moveHandler`. Le paramètre d'événement permet à la fonction de gestionnaire d'examiner l'objet événement. Dans cet exemple, vous utilisez la propriété `type` de l'objet événement pour indiquer si cet événement est de type `move`.

Suppression des écouteurs d'événement

Adobe AIR 1.0 et les versions ultérieures

La méthode `removeEventListener()` permet de supprimer un écouteur d'événement dont vous n'avez plus besoin. Il est judicieux de supprimer tous les écouteurs qui ne seront plus utilisés. Les paramètres requis sont notamment `eventName` et `listener`, soit les mêmes que ceux requis pour la méthode `addEventListener()`.

Suppression d'écouteurs d'événement sur une page HTML de navigation

Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un contenu HTML navigue ou qu'il est éliminé parce qu'une fenêtre dans laquelle il figure est fermée, les écouteurs d'événement qui font référence aux objets sur la page déchargée ne sont pas automatiquement supprimés. Lorsqu'un objet distribue un événement à un gestionnaire qui a déjà été déchargé, le message d'erreur suivant s'affiche : « The application attempted to reference a JavaScript object in an HTML page that is no longer loaded » (L'application a tenté de référencer un objet JavaScript sur une page HTML qui n'est plus chargée).

Pour éviter que ce message d'erreur ne s'affiche, supprimez les écouteurs d'événement JavaScript intégrés à une page HTML avant qu'elle ne soit déchargée. En cas de navigation de page (au sein d'un objet HTMLLoader), supprimez l'écouteur d'événement lors de l'événement `unload` de l'objet `window`.

Par exemple, le code JavaScript suivant supprime un écouteur associé à un événement `uncaughtScriptException` :

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Pour empêcher que cette erreur ne se produise à la fermeture des fenêtres comportant un contenu HTML, appelez une fonction de nettoyage en réponse à l'événement `closing` de l'objet NativeWindow (`window.nativeWindow`). Par exemple, le code JavaScript suivant supprime un écouteur associé à un événement `uncaughtScriptException` :

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Pour éviter que cette erreur ne se produise, vous pouvez également supprimer un écouteur d'événement dès son exécution (sous réserve que l'événement ne doive être géré qu'une seule fois). Par exemple, le code JavaScript suivant crée une fenêtre HTML en appelant la méthode `createRootWindow()` de la classe HTMLLoader et ajoute un écouteur associé à l'événement `complete`. Lorsque le gestionnaire d'événement `complete` est appelé, il supprime son propre écouteur d'événement par le biais de la fonction `removeEventListener()` :

```
 var html = runtime.flash.html.HTMLLoader.createRootWindow(true);
html.addEventListener('complete', htmlCompleteListener);
function htmlCompleteListener()
{
    html.removeEventListener(complete, arguments.callee)
    // handler code..
}
html.load(new runtime.flash.net.URLRequest("second.html"));
```

La suppression d'écouteurs d'événement superflus permet également au nettoyeur de mémoire système de récupérer la mémoire éventuellement mobilisée par ces écouteurs.

Vérification des écouteurs d'événement existants

Adobe AIR 1.0 et les versions ultérieures

La méthode `hasEventListener()` vous permet de vérifier l'existence d'un écouteur d'événement associé à un objet.

Chapitre 62 : Affichage de contenu HTML dans une application mobile

Adobe AIR 2.5 et les versions ultérieures

La classe StageWebView affiche un contenu HTML à l'aide du contrôle de navigateur système sur un périphérique mobile et du contrôle HTMLLoader standard d'Adobe® AIR® sur un poste de travail. Vérifiez la propriété `StageWebView.isSupported` pour déterminer si la classe est prise en charge sur le périphérique en cours. Le profil mobile n'assure pas la prise en charge de tous les périphériques.

Dans tous les profils, la classe StageWebView ne prend en charge qu'une interaction restreinte entre le contenu HTML et le reste de l'application. Vous pouvez contrôler la navigation, mais l'intercodage ou les échanges directs de données sont interdits. Vous pouvez charger un contenu à partir d'une URL locale ou distante ou transmettre une chaîne de contenu HTML.

Objets StageWebView

Un objet StageWebView n'est pas un objet d'affichage et il est impossible de l'ajouter à la liste d'affichage. Il fait plutôt office de fenêtre d'affichage directement associée à la scène. Un contenu StageWebView est superposé à n'importe quel contenu de liste d'affichage. Il n'existe aucun moyen de contrôler l'ordre de tracé de plusieurs objets StageWebView.

Pour afficher un objet StageWebView, vous affectez la scène sur laquelle il doit apparaître à sa propriété `stage`. Définissez la taille de l'affichage à l'aide de la propriété `viewPort`.

Définissez les coordonnées x et y de la propriété `viewPort` sur une valeur comprise entre -8192 et 8191. La largeur et la hauteur maximales de la scène ne doivent pas dépasser 8191. Si la taille excède les valeurs maximales, une exception est renvoyée.

L'exemple suivant crée un objet StageWebView, définit les propriétés `stage` et `viewPort` et affiche une chaîne de contenu HTML :

```
var webView:StageWebView = new StageWebView();
webView.viewPort = new Rectangle( 0, 0, this.stage.stageWidth, this .stage.stageHeight);
webView.stage = this.stage;
var htmlString:String = "<!DOCTYPE HTML>" +
                        "<html><body>" +
                        "<p>King Philip could order five good steaks.</p>" +
                        "</body></html>";
webView.loadString( htmlString );
```

Pour masquer un objet StageWebView, définissez la propriété `stage` correspondante sur `null`. Pour détruire complètement l'objet, appelez la méthode `dispose()`. Bien que l'appel de la méthode `dispose()` soit facultatif, il permet au nettoyeur de mémoire de récupérer plus tôt la mémoire utilisée par l'objet.

Contenu

Vous disposez de deux méthodes, `loadURL()` et `loadString()`, pour charger un contenu dans un objet StageWebView.

La méthode `loadURL()` charge une ressource à l'adresse URL indiquée. Libre à vous de faire appel à tout modèle d'URI pris en charge par le contrôle du navigateur Web du système, tel que data:, file:, http:, https: et javascript:. Les modèles app: et app-storage: ne sont pas pris en charge. AIR ne valide pas la chaîne URL.

La méthode `loadString()` charge une chaîne littérale qui intègre un contenu HTML. L'emplacement d'une page chargée par le biais de cette méthode est exprimé comme suit :

- Sur un système d'exploitation de bureau : about:blank
- Sur iOS : *htmlString*
- Sur Android : format URI de données de l'élément *htmlString* codé

Le modèle d'URI détermine les règles de chargement de contenu ou de données intégrés.

Modèle d'URI	Chargement de ressource locale	Chargement de ressource distante	Objet XMLHttpRequest local	Objet XMLHttpRequest distant
data:	Non	Oui	Non	Non
file:	Oui	Oui	Oui	Oui
http:, https:	Non	Oui	Non	Même domaine
about: (méthode loadString())	Non	Oui	Non	Non

Remarque : *si la propriété `displayState` de la scène est définie sur `FULL_SCREEN`, il est impossible de renseigner sur Desktop un champ de texte affiché dans l'objet StageWebView. Toutefois, sur iOS et Android, vous pouvez saisir des données dans un champ de texte de l'objet StageWebView même si la propriété `displayState` de la scène est définie sur `FULL_SCREEN`.*

L'exemple suivant fait appel à l'objet StageWebView pour afficher le site Web d'Adobe :

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;

    public class StageWebViewExample extends MovieClip{

        var webView:StageWebView = new StageWebView();

        public function StageWebViewExample() {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadURL( "http://www.adobe.com" );
        }
    }
}
```

Sur un périphérique Android, vous devez spécifier l'autorisation INTERNET Android pour permettre à l'application de charger des ressources distantes.

Dans Android 3.0+, l'application doit activer l'accélération matérielle dans l'élément manifestAdditions Android du descripteur d'application AIR pour afficher le contenu des modules d'extension dans un objet StageWebView. Voir Activation de Flash Player et d'autres modules d'extension dans un objet StageWebView.

URI JavaScript

Vous disposez d'un URI JavaScript pour appeler une fonction définie sur la page HTML chargée par un objet StageWebView. La fonction appelée par le biais de l'URI JavaScript s'exécute dans le contexte de la page Web chargée. L'exemple suivant appelle une fonction JavaScript à l'aide d'un objet StageWebView :

```
package {
    import flash.display.*;
    import flash.geom.Rectangle;
    import flash.media.StageWebView;
    public class WebView extends Sprite
    {
        public var webView:StageWebView = new StageWebView();
        public function WebView()
        {
            var htmlString:String = "<!DOCTYPE HTML>" +
            "<html><script type=text/javascript>" +
            "function callURI(){" +
            "alert(\"You clicked me!!\");"+
            "}</script><body>" +
            "<p><a href=javascript:callURI()>Click Me</a></p>" +
            "</body></html>";
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadString( htmlString );
        }
    }
}
```

Voir aussi

Sean Voisen : Making the Most of StageWebView (disponible en anglais uniquement)

Evénements de navigation

Lorsqu'un utilisateur clique sur un lien dans HTML, l'objet StageWebView distribue un événement `locationChanging`. Vous pouvez appeler la méthode `preventDefault()` de l'objet d'événement pour arrêter la navigation. Si tel n'est pas le cas, l'objet StageWebView navigue jusqu'à la page suivante et distribue un événement `locationChange`. Une fois le chargement de la page terminé, l'objet StageWebView distribue un événement `complete`.

Un événement `locationChanging` est distribué lors de chaque redirection HTML. Les événements `locationChange` et `complete` sont distribués au moment opportun.

Sur iOS, un événement `locationChanging` est distribué avant un événement `locationChange`, à l'exception des premières méthodes `loadURL()` ou `loadString()`. La distribution d'un événement `locationChange` est également associée aux changements de navigation via des iFrames et images.

L'exemple suivant illustre la procédure à mettre en œuvre pour interdire un changement d'emplacement et ouvrir la nouvelle page dans le navigateur système.

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.events.LocationChangeEvent;
    import flash.geom.Rectangle;
    import flash.net.navigateToURL;
    import flash.net.URLRequest;

    public class StageWebViewNavEvents extends MovieClip{
        var webView:StageWebView = new StageWebView();

        public function StageWebViewNavEvents() {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
        webView.addEventListener( LocationChangeEvent.LOCATION_CHANGING, onLocationChanging );
            webView.loadURL( "http://www.adobe.com" );
        }
        private function onLocationChanging( event:LocationChangeEvent ):void
        {
            event.preventDefault();
            navigateToURL( new URLRequest( event.location ) );
        }
    }
}
```

Historique

Lorsqu'un utilisateur clique dans le contenu affiché dans un objet StageWebView, le contrôle enregistre les piles d'historique en amont et en aval. L'exemple suivant illustre la procédure à mettre en œuvre pour naviguer d'une pile d'historique à l'autre. L'exemple fait appel aux touches programmables Back et Search.

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;

    public class StageWebViewExample extends MovieClip{

        var webView:StageWebView = new StageWebView();

        public function StageWebViewExample()
        {
            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 0, stage.stageWidth, stage.stageHeight );
            webView.loadURL( "http://www.adobe.com" );

            stage.addEventListener( KeyboardEvent.KEY_DOWN, onKey );
        }

        private function onKey( event:KeyboardEvent ):void
        {
            if( event.keyCode == Keyboard.BACK && webView.isHistoryBackEnabled )
            {
                trace("back");
                webView.historyBack();
                event.preventDefault();
            }
            if( event.keyCode == Keyboard.SEARCH && webView.isHistoryForwardEnabled )
            {
                trace("forward");
                webView.historyForward();
            }
        }
    }
}
```

Cible d'action

Bien qu'il ne s'agisse pas d'un objet d'affichage, la classe StageWebView contient des membres destinés à gérer les transitions de cible d'action par rapport au contrôle.

Lorsque l'objet StageWebView devient la cible d'action, il distribue un événement `focusIn`. Le cas échéant, cet événement permet de gérer les éléments cibles d'action de l'application.

Lorsque l'objet StageWebView cesse d'être la cible d'action, il distribue un événement `focusOut`. Une occurrence de l'objet StageWebView peut cesser d'être la cible d'action lorsqu'un utilisateur « évite » le premier ou le dernier contrôle de la page Web à l'aide des touches de direction ou d'un trackball (boule roulante) de périphérique. La propriété `direction` de l'objet d'événement permet de savoir si la cible d'action remonte au-delà du haut de la page ou descend via le bas de la page. Grâce à ces informations, vous pouvez faire de l'objet d'affichage approprié placé au-dessus ou au-dessous de l'objet StageWebView la cible d'action.

Sur iOS, il est impossible de définir la cible d'action par programmation. L'objet StageWebView distribue les événements `focusIn` et `focusOut` en définissant la propriété de direction de `FocusEvent` sur `none`. Si l'utilisateur touche l'intérieur de l'objet StageWebView, l'événement `focusIn` est distribué. Si l'utilisateur touche l'extérieur de l'objet StageWebView, l'événement `focusOut` est distribué.

L'exemple suivant illustre la transition de la cible d'action de l'objet StageWebView aux objets d'affichage Flash :

```
package  {
    import flash.display.MovieClip;
    import flash.media.StageWebView;
    import flash.geom.Rectangle;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    import flash.text.TextField;
    import flash.text.TextFieldType;
    import flash.events.FocusEvent;
    import flash.display.FocusDirection;
    import flash.events.LocationChangeEvent;

    public class StageWebViewFocusEvents extends MovieClip{
        var webView:StageWebView = new StageWebView();
        var topControl:TextField = new TextField();
        var bottomControl:TextField = new TextField();

        public function StageWebViewFocusEvents()
        {
            trace("Starting");
            topControl.type = TextFieldType.INPUT;
            addChild( topControl );
            topControl.height = 60;
            topControl.width = stage.stageWidth;
            topControl.background = true;
            topControl.text = "One control on top.";
            topControl.addEventListener( FocusEvent.FOCUS_IN, flashFocusIn );
            topControl.addEventListener( FocusEvent.FOCUS_OUT, flashFocusOut );

            webView.stage = this.stage;
            webView.viewPort = new Rectangle( 0, 60, stage.stageWidth, stage.stageHeight
- 120 );
            webView.addEventListener( FocusEvent.FOCUS_IN, webFocusIn );
            webView.addEventListener(FocusEvent.FOCUS_OUT, webFocusOut );
            webView.addEventListener(LocationChangeEvent.LOCATION_CHANGING,
                                     function( event:LocationChangeEvent ):void
                                     {
                                        event.preventDefault();
                                     } );
            webView.loadString("<form action='#'><input/><input/><input/></form>");
            webView.assignFocus();

            bottomControl.type = TextFieldType.INPUT;
            addChild( bottomControl );
            bottomControl.y = stage.stageHeight - 60;
            bottomControl.height = 60;
            bottomControl.width = stage.stageWidth;
            bottomControl.background = true;
            bottomControl.text = "One control on the bottom.";
            bottomControl.addEventListener( FocusEvent.FOCUS_IN, flashFocusIn );
```

```
            bottomControl.addEventListener( FocusEvent.FOCUS_OUT, flashFocusOut );}

        private function webFocusIn( event:FocusEvent ):void
        {
            trace("Web focus in");
        }

        private function webFocusOut( event:FocusEvent ):void
        {
            trace("Web focus out: " + event.direction);
            if( event.direction == FocusDirection.TOP )
            {
                stage.focus = topControl;
            }
            else
            {
                stage.focus = bottomControl;
            }
        }

        private function flashFocusIn( event:FocusEvent ):void
        {
            trace("Flash focus in");
            var textfield:TextField = event.target as TextField;
            textfield.backgroundColor = 0xff5566;
        }

        private function flashFocusOut( event:FocusEvent ):void
        {
            trace("Flash focus out");
            var textfield:TextField = event.target as TextField;
            textfield.backgroundColor = 0xffffff;
        }

    }
}
```

Capture d'image bitmap

Un objet StageWebView est superposé au contenu de la liste d'affichage. Vous ne pouvez pas ajouter de contenu par-dessus un objet StageWebView. Il est, par exemple, impossible de développer une liste déroulante par-dessus le contenu StageWebView. Pour résoudre ce problème, capturez un instantané de l'objet StageWebView. Masquez ensuite l'objet StageWebView et ajoutez à sa place l'instantané de l'image bitmap.

L'exemple suivant illustre la capture de l'instantané d'un objet StageWebView à l'aide de la méthode `drawViewPortToBitmapData`. Pour masquer l'objet StageWebView, la scène est définie sur null. Une fois le chargement complet de la page Web terminé, la fonction appelée capture l'image bitmap et l'affiche. A l'exécution, le code affiche deux libellés, Google et Facebook. Le fait de cliquer sur l'étiquette capture la page Web correspondante et l'affiche sur la scène sous la forme d'un instantané.

```
package
{
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.geom.Rectangle;
    import flash.media.StageWebView;
    import flash.net.*;
    import flash.text.TextField;
    public class stagewebview extends Sprite
    {
        public var webView:StageWebView=new StageWebView();
        public var textGoogle:TextField=new TextField();
        public var textFacebook:TextField=new TextField();
        public function stagewebview()
        {
            textGoogle.htmlText="<b>Google</b>";
            textGoogle.x=300;
            textGoogle.y=-80;
            addChild(textGoogle);
            textFacebook.htmlText="<b>Facebook</b>";
            textFacebook.x=0;
            textFacebook.y=-80;
            addChild(textFacebook);
            textGoogle.addEventListener(MouseEvent.CLICK,goGoogle);
            textFacebook.addEventListener(MouseEvent.CLICK,goFaceBook);
            webView.stage = this.stage;
            webView.viewPort = new Rectangle(0, 0, stage.stageWidth, stage.stageHeight);
        }
    public function goGoogle(e:Event):void
        {
            webView.loadURL("http://www.google.com");
            webView.stage = null;
            webView.addEventListener(Event.COMPLETE,handleLoad);
        }

    public function goFaceBook(e:Event):void
        {
            webView.loadURL("http://www.facebook.com");
            webView.stage = null;
            webView.addEventListener(Event.COMPLETE,handleLoad);
        }
    public function handleLoad(e:Event):void
        {
            var bitmapData:BitmapData = new BitmapData(webView.viewPort.width,
webView.viewPort.height);
            webView.drawViewPortToBitmapData(bitmapData);
            var webViewBitmap:Bitmap=new Bitmap(bitmapData);
            addChild(webViewBitmap);
        }
    }
}
```

Chapitre 63 : Utilisation de programmes de travail à des fins de simultanéité

Flash Player 11.4 et les versions ultérieures, Adobe AIR 13.4 et les versions ultérieures pour les plates-formes de bureau

Grâce aux programmes de travail ActionScript, il est possible d'exécuter du code simultanément, c'est-à-dire d'exécuter du code en arrière-plan sans interrompre l'exécution du code principal.

Les API de simultanéité d'ActionScript sont disponibles sur les plates-formes de bureau uniquement dans Flash Player 11.4 et les versions ultérieures et dans AIR 3.4 et les versions ultérieures. La simultanéité n'est pas prise en charge dans AIR sur les plates-formes mobiles.

Présentation des programmes de travail et de la simultanéité

Flash Player 11.4 et les versions ultérieures, Adobe AIR 13.4 et les versions ultérieures pour les plates-formes de bureau

Lorsqu'une application n'utilise pas des programmes de travail, son code s'exécute dans un bloc d'exécution linéaire unique appelé *thread* d'exécution. Ce thread exécute le code écrit par un développeur. Il exécute également une grande partie du code appartenant au moteur d'exécution, plus particulièrement le code qui met à jour l'écran lorsque les propriétés des objets d'affichage changent. Bien que le code soit écrit par blocs en tant que méthodes et classes, il s'exécute ligne par ligne au moment de l'exécution même s'il a été écrit dans une longue série d'étapes. Supposons qu'une application exécute les étapes suivantes :

1. Image d'entrée : le moteur d'exécution appelle des gestionnaires d'événement `enterFrame` et exécute tour à tour leur code.
2. Evénement de souris : l'utilisateur déplace la souris, et le moteur d'exécution appelle les gestionnaires d'événements de souris à mesure que se produisent les événements de survol et de déroulement.
3. Evénement de fin de chargement : une demande de chargement d'un fichier xml provenant d'une URL est renvoyée avec les données du fichier chargé. Le gestionnaire d'événement est appelé et exécute ses étapes en lisant le contenu xml et en créant un ensemble d'objets à partir des données xml.
4. Evénement de souris : la souris s'est à nouveau déplacée ; le moteur d'exécution appelle donc les gestionnaires d'événements de souris pertinents.
5. Rendu : plus aucun événement n'est en attente ; le moteur d'exécution met donc à jour l'écran en fonction des modifications qu'ont subi les objets d'affichage.
6. Image d'entrée : le cycle recommence.

Comme le montre cet exemple, les étapes 1 à 5 s'exécutent en séquence au sein d'un bloc temporel unique appelé image. Etant donné qu'elles s'exécutent en séquence dans un thread unique, le moteur d'exécution ne peut pas interrompre une étape du processus pour en exécuter une autre. A une cadence de 30 images par seconde, le moteur d'exécution dispose de moins d'un trentième de seconde pour exécuter l'ensemble de ces opérations. En règle générale, ce délai est suffisant pour exécuter le code et le moteur d'exécution n'a plus qu'à attendre pendant le temps restant. Supposez néanmoins que les données xml qui se chargent à l'étape 3 représentent une très grande structure xml profondément imbriquée. Etant donné que le code s'exécute en boucle sur les données xml et crée des objets, il est tout à fait possible que cette opération prenne plus d'un trentième de seconde. Dans ce cas, les dernières étapes (répondre à la souris et actualiser l'écran) ne sont pas exécutées au moment où elles le devraient. Résultat : l'écran se bloque, car il n'est pas actualisé assez vite en réponse à l'utilisateur qui déplace la souris.

Si l'ensemble du code s'exécute dans le même thread, il n'existe qu'un seul moyen d'éviter les blocages occasionnels : éviter les opérations longues telles que l'exécution en boucle sur un jeu de données volumineux. Les programmes de travail d'ActionScript proposent une solution alternative. Un programme de travail peut vous permettre d'exécuter du code à exécution longue dans un programme de travail indépendant. Comme chaque programme de travail est exécuté dans un thread différent, le programme de travail en arrière-plan lance l'opération longue dans son propre thread. Cela évite ainsi au thread d'exécution du programme de travail principal d'actualiser l'écran à chaque image et d'être bloqué par un autre processus.

La possibilité d'exécuter plusieurs opérations de code en même temps de cette manière est appelée *simultanéité*. Lorsque le programme de travail en arrière-plan finit son travail, ou à certaines étapes-clés du processus, vous pouvez envoyer au programme de travail principal des notifications et des données. De cette manière, vous pouvez écrire du code qui exécute des opérations longues ou complexes tout en évitant le blocage de l'écran.

Les programmes de travail sont utiles, car ils diminuent le risque d'une baisse de la cadence, qui peut se produire si le thread de rendu principal est bloqué par l'exécution d'un autre code. Les programmes de travail augmentent toutefois l'utilisation de la mémoire système et du processeur, ce qui peut avoir une incidence considérable sur les performances globales de l'application. Etant donné que chaque programme de travail utilise sa propre occurrence de la machine virtuelle du moteur d'exécution, le traitement d'un programme de travail, aussi simple soit-il, peut consommer de nombreuses ressources. Lorsque vous utilisez des programmes de travail, testez votre code sur toutes vos plates-formes cibles afin de vous assurer que les ressources requises ne sont pas trop importantes. Adobe vous recommande de ne pas utiliser plus de deux programmes de travail dans un scénario type.

Création et gestion de programmes de travail

Flash Player 11.4 et les versions ultérieures, Adobe AIR 13.4 et les versions ultérieures pour les plates-formes de bureau

La première étape pour utiliser un programme de travail à des fins de simultanéité est de créer un programme de travail en arrière-plan. Vous devez utiliser deux types d'objets pour créer un programme de travail. Une occurrence de l'objet Worker, que vous créez. Un objet WorkerDomain, qui crée le programme de travail et gère les objets Worker qui s'exécutent dans une application.

Lorsque le moteur d'exécution se charge, il crée l'objet WorkerDomain. Par ailleurs, le moteur d'exécution crée automatiquement un programme de travail pour le fichier swf de l'application. Ce premier programme de travail est appelé *programme de travail primordial*.

Etant donné qu'il n'existe qu'un seul objet WorkerDomain pour une application, vous accédez à l'occurrence de l'objet WorkerDomain à l'aide de la propriété statique `WorkerDomain.current`.

Vous pouvez à tout moment accéder à l'occurrence de l'objet Worker actuel (le programme de travail dans lequel le code actuel s'exécute) à l'aide de la propriété statique `Worker.current`.

Création d'un objet Worker à partir d'un fichier swf

Tout comme le fichier swf principal qui s'exécute au sein du programme de travail primordial, un programme de travail en arrière-plan exécute le code d'un seul fichier swf. Pour utiliser un programme de travail en arrière-plan, vous devez créer et compiler le code du programme de travail dans un fichier swf. Pour créer le programme de travail en arrière-plan, le programme de travail parent doit accéder aux octets de ce fichier swf en tant qu'objet ByteArray. Vous devez transmettre cet objet ByteArray à la méthode `createWorker()` de l'objet WorkerDomain pour créer le programme de travail.

Il existe trois méthodes principales pour convertir le fichier swf de programme de travail en arrière-plan en objet ByteArray :

Intégration du fichier swf de programme de travail

Utilisez la métabalise [Embed] pour intégrer le fichier swf de programme de travail dans le fichier swf principal sous forme d'objet ByteArray :

```
[Embed(source="../swfs/BgWorker.swf", mimeType="application/octet-stream")]
private static var BgWorker_ByteClass:Class;
private function createWorker():void
{
    var workerBytes:ByteArray = new BgWorker_ByteClass();
    var bgWorker:Worker = WorkerDomain.current.createWorker(workerBytes);

    // ... set up worker communication and start the worker
}
```

Le fichier swf de programme de travail est compilé dans le fichier swf principal sous forme de sous-classe ByteArray appelée BgWorker_ByteClass. Créez une occurrence de cette classe pour obtenir un objet ByteArray prérempli avec les octets du fichier swf de programme de travail.

Chargement d'un fichier swf de programme de travail externe

Utilisez un objet URLLoader pour charger un fichier swf externe. Le fichier swf doit provenir du même domaine de sécurité ; il peut s'agir, par exemple, d'un fichier swf chargé depuis le même domaine Internet que le fichier swf principal ou inclus dans le package d'une application AIR.

```
var workerLoader:URLLoader = new URLLoader();
workerLoader.dataFormat = URLLoaderDataFormat.BINARY;
workerLoader.addEventListener(Event.COMPLETE, loadComplete);
workerLoader.load(new URLRequest("BgWorker.swf"));

private function loadComplete(event:Event):void
{
    // create the background worker
    var workerBytes:ByteArray = event.target.data as ByteArray;
    var bgWorker:Worker = WorkerDomain.current.createWorker(workerBytes);

    // ... set up worker communication and start the worker
}
```

Une fois que l'objet URLLoader termine le chargement du fichier swf, les octets du fichier swf sont disponibles dans la propriété `data` de l'objet URLLoader (`event.target.data` dans l'exemple).

Utilisation du fichier swf principal comme fichier swf de programme de travail

Vous pouvez utiliser un seul fichier swf comme fichier swf principal et fichier swf de programme de travail. Utilisez la propriété `loaderInfo.bytes` de la classe d'affichage principale pour accéder aux octets du fichier swf.

```
// The primordial worker's main class constructor
public function PrimordialWorkerClass()
{
    init();
}

private function init():void
{
    var swfBytes:ByteArray = this.loaderInfo.bytes;

    // Check to see if this is the primordial worker or the background worker
    if (Worker.current.isPrimordial)
    {
        // create a background worker
        var bgWorker:Worker = WorkerDomain.current.createWorker(swfBytes);

        // ... set up worker communication and start the worker
    }
    else // entry point for the background worker
    {
        // set up communication between workers using getSharedProperty()
        // ... (not shown)

        // start the background work
    }
}
```

Si vous utilisez cette technique, faites appel à une déclaration `if` pour relier le code du fichier swf au sein du constructeur de la classe principale ou d'une méthode qu'il appelle. Pour déterminer si le code s'exécute dans le programme de travail principal ou dans le programme de travail en arrière-plan, vérifiez la propriété `isPrimordial` de l'objet Worker actuel, comme indiqué dans l'exemple.

Lancement de l'exécution d'un programme de travail

Une fois le programme de travail créé, lancez l'exécution de son code en appelant la méthode `start()` de l'objet Worker. L'opération `start()` ne commence pas immédiatement. Pour savoir quand le programme de travail est exécuté, enregistrez un écouteur pour l'événement `workerState` de l'objet Worker. Cet événement est distribué lorsque l'objet Worker change d'état au cours de son cycle de vie, notamment lorsqu'il commence à exécuter le code. Dans votre gestionnaire d'événement `workerState`, vérifiez que la propriété `state` de l'objet Worker est définie sur `WorkerState.RUNNING`. A ce stade, le programme de travail est en cours d'exécution et le constructeur de sa classe principale s'est exécuté. Consultez la liste de code suivante pour obtenir un exemple d'enregistrement de l'événement `workerState` et d'appel de la méthode `start()` :

```
// listen for worker state changes to know when the worker is running
bgWorker.addEventListener(Event.WORKER_STATE, workerStateHandler);
// set up communication between workers using
// setSharedProperty(), createMessageChannel(), etc.
// ... (not shown)
bgWorker.start();
private function workerStateHandler(event:Event):void
{
    if (bgWorker.state == WorkerState.RUNNING)
    {
        // The worker is running.
        // Send it a message or wait for a response.
    }
}
```

Gestion de l'exécution du programme de travail

Vous pouvez à tout moment accéder à l'ensemble des programmes de travail en cours d'exécution dans votre application à l'aide de la méthode `listWorkers()` de la classe WorkerDomain. Cette méthode renvoie l'ensemble des programmes de travail dont la propriété `state` est définie sur `WorkerState.RUNNING`, y compris le programme de travail primordial. Si un programme de travail n'a pas démarré ou si son exécution a été arrêtée, il n'est pas inclus.

Si vous n'avez plus besoin d'un programme de travail, vous pouvez appeler la méthode `terminate()` de l'objet Worker pour arrêter le programme de travail et libérer sa mémoire, ainsi que les autres ressources du système.

Communication entre programmes de travail

Flash Player 11.4 et les versions ultérieures, Adobe AIR 13.4 et les versions ultérieures pour les plates-formes de bureau

Bien que les programmes de travail exécutent leur code dans des threads d'exécution distincts, ils n'auraient aucun intérêt s'ils étaient totalement isolés les uns des autres. La communication entre les programmes de travail signifie principalement échanger des données. Il existe trois mécanismes principaux pour transmettre des données d'un programme de travail à un autre.

Au moment de choisir la technique de partage la mieux adaptée à vos besoins de transmission de données, tenez compte des éléments qui les différencient. La première différence entre ces techniques est de savoir s'il existe un événement informant le récepteur que de nouvelles données sont disponibles ou si le programme de travail de réception doit rechercher les mises à jour. La deuxième différence entre ces techniques de partage de données porte sur la méthode de transmission des données. Dans certains cas, le programme de travail de réception obtient une copie des données partagées, ce qui implique la création de davantage d'objets, et donc une plus grande utilisation de la mémoire et du processeur. Dans d'autres cas, les programmes de travail accèdent aux objets faisant référence à la même mémoire système sous-jacente, ce qui implique la création de moins d'objets, et donc une plus faible utilisation de la mémoire. Ces différences sont précisées ici :

Technique de communication	Distribution de l'événement à la réception des données	Partage de la mémoire entre programmes de travail
Propriétés partagées entre programmes de travail	Non	Non, les objets sont des copies et non des références
MessageChannel	Oui	Non, les objets sont des copies et non des références
Objet ByteArray partageable	Non	Oui, la mémoire est partagée

Transmission de données avec une propriété partagée

La méthode la plus simple pour partager des données entre programmes de travail est d'utiliser une propriété partagée. Chaque programme de travail conserve un dictionnaire interne de valeurs de propriété partagée. Pour distinguer les propriétés entre elles, celles-ci sont enregistrées avec des noms de clés de type chaîne. Pour enregistrer un objet sur un programme de travail en tant que propriété partagée, appelez la méthode `setSharedProperty()` de l'objet Worker avec deux arguments, le nom de clé et la valeur à enregistrer :

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

Une fois la propriété partagée définie, il est possible de lire la valeur en appelant la méthode `getSharedProperty()` de l'objet Worker en transmettant le nom de clé :

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

N'importe quel programme de travail peut lire ou définir la valeur de la propriété. Par exemple, le code dans un programme de travail en arrière-plan peut appeler sa méthode `setSharedProperty()` pour enregistrer une valeur. Le code qui s'exécute dans le programme de travail parent peut alors faire appel à la propriété `getSharedProperty()` pour recevoir les données.

Pratiquement tout type d'objet peut être transmis comme valeur à la méthode `setSharedProperty()`. Lorsque vous appelez la méthode `getSharedProperty()`, l'objet renvoyé est une copie de l'objet transmis à la méthode `setSharedProperty()` et non une référence au même objet, sauf dans quelques cas isolés. La méthode de partage des données est expliquée en détails à la section « Références partagées et valeurs copiées » à la page 1083.

Le principal avantage d'utiliser une propriété partagée pour transmettre des données entre programmes de travail est que celle-ci est disponible même avant que le programme de travail ne commence à s'exécuter. Vous pouvez appeler la méthode `setSharedProperty()` d'un objet de programme de travail en arrière-plan pour définir une propriété partagée, même avant que le programme de travail ne commence à s'exécuter. Lorsque le programme de travail parent appelle la méthode `start()` de l'objet Worker, le moteur d'exécution appelle le constructeur de la classe principale du programme de travail enfant. Toutes les propriétés partagées définies avant l'appel de la méthode `start()` sont disponibles dans le code afin que le programme de travail enfant puisse les lire.

Transmission de données via un canal de message

Un canal de message fournit un lien de transmission de données unidirectionnel entre deux programmes de travail. L'utilisation d'un objet MessageChannel pour transmettre des données entre programmes de travail présente un avantage majeur. Lorsque vous envoyez un message (un objet) à l'aide d'un canal de message, l'objet MessageChannel distribue un événement `channelMessage`. Le code dans le programme de travail de réception peut écouter cet événement pour savoir quand les données seront disponibles. De cette façon, le programme de travail de réception n'a pas besoin de rechercher continuellement des mises à jour de données.

Un canal de message est associé à deux programmes de travail uniquement : un programme de travail d'envoi et un programme de travail de réception. Pour créer un objet MessageChannel, appelez la méthode `createMessageChannel()` de l'objet Worker d'envoi, en transmettant le programme de travail de réception en tant qu'argument :

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

Les deux programmes de travail doivent avoir accès à l'objet MessageChannel. La méthode la plus simple pour effectuer cette opération est de transmettre l'objet MessageChannel à l'aide de la méthode `setSharedProperty()` :

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

Dans le programme de travail de réception, enregistrez un écouteur pour l'événement `channelMessage` de l'objet MessageChannel. Cet événement est distribué lorsque le programme de travail d'envoi envoie les données via le canal de message.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

Pour envoyer les données, appelez la méthode `send()` de l'objet MessageChannel dans le programme de travail d'envoi :

```
// In the sending worker swf
sendChannel.send("This is a message");
```

Dans le programme de travail de réception, l'objet MessageChannel appelle l'écouteur d'événement `channelMessage`. Le programme de travail de réception peut alors obtenir les données en appelant la méthode `receive()` de l'objet MessageChannel.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

L'objet renvoyé par la méthode de réception possède le même type de données que l'objet transmis dans la méthode `send()`. L'objet reçu est une copie de l'objet transmis par le programme de travail d'envoi et non une référence à l'objet dans le programme de travail de réception, à moins qu'il ne soit de l'un des types de données pris en charge, comme décrit à la section « Références partagées et valeurs copiées » à la page 1083.

Partage de données avec un objet ByteArray partageable

Lors de la transmission d'un objet entre deux programmes de travail, le programme de travail de réception obtient un nouvel objet, qui est une copie de l'objet original. Ces deux objets sont stockés à divers emplacements dans la mémoire du système. Chaque copie de l'objet qui est reçue augmente donc la mémoire totale utilisée par le moteur d'exécution. Par ailleurs, les modifications apportées à un objet dans un programme de travail ne sont pas appliquées à la copie dans l'autre programme de travail. Pour plus d'informations sur la méthode de copie des objets, voir « Références partagées et valeurs copiées » à la page 1083.

Par défaut, un objet ByteArray se comporte de la même façon. Si vous transmettez une occurrence de ByteArray à la méthode `setSharedProperty()` de l'objet Worker ou à la méthode `send()` de l'objet MessageChannel, le moteur d'exécution crée un nouvel objet ByteArray dans la mémoire de l'ordinateur et le programme de travail de réception obtient une occurrence de ByteArray qui est une référence au nouvel objet ByteArray. Vous pouvez néanmoins changer ce comportement pour un objet ByteArray en définissant sa propriété `shareable` sur `true`.

Lorsqu'un objet ByteArray partageable est transmis d'un programme de travail à un autre, l'occurrence de ByteArray dans le programme de travail de réception est une référence à la même mémoire système sous-jacente utilisée par l'occurrence de ByteArray dans le programme de travail d'envoi. Lorsque le code d'un programme de travail modifie le contenu du tableau d'octets, ces modifications sont immédiatement disponibles dans d'autres programmes de travail ayant accès à ce tableau d'octets partagé.

Etant donné que les programmes de travail exécutent leur code simultanément, deux programmes de travail peuvent tenter d'accéder en même temps aux mêmes octets dans un tableau d'octets, ce qui pourrait entraîner la perte ou la corruption des données. Vous pouvez utiliser plusieurs API pour gérer l'accès aux ressources partagées et éviter ces problèmes.

La classe ByteArray dispose de méthodes vous permettant de valider et de modifier le contenu du tableau d'octets en une seule opération :

- Méthode atomicCompareAndSwapIntAt()
- Méthode atomicCompareAndSwapLength()

En outre, le package flash.concurrent inclut des classes qui permettent de contrôler l'accès aux ressources partagées :

- Classe Mutex
- Classe Condition

Références partagées et valeurs copiées

En temps normal, lorsque vous appelez la méthode `Worker.setSharedProperty()` ou la méthode `MessageChannel.send()`, l'objet transmis au programme de travail de réception est transmis par sérialisation au format AMF. Cela entraîne certaines conséquences :

- L'objet créé dans le programme de travail de réception lors de l'appel de sa méthode `getSharedProperty()` est désérialisé des octets AMF. Il est une copie de l'objet original et non une référence à l'objet. Toute modification apportée à l'objet dans l'un des programmes de travail n'est pas appliquée à la copie dans l'autre programme de travail.
- Les objets qu'il est impossible de sérialiser au format AMF, notamment les objets d'affichage, ne peuvent pas être transmis à un programme de travail qui utilise `Worker.setSharedProperty()` ou `MessageChannel.send()`.
- Pour désérialiser correctement une classe personnalisée, la définition de classe doit être enregistrée avec la fonction `flash.net.registerClassAlias()` ou les métadonnées `[RemoteClass]`. Le même alias doit être utilisé pour les deux versions du programme de travail de la classe .

Il existe cinq types d'objets véritablement partagés au lieu d'être copiés entre programmes de travail :

- Objets Worker
- Objets MessageChannel
- Tableau d'octets partageable (objet ByteArray dont la propriété `shareable` est définie sur `true`)
- Objets Mutex
- Objets Condition

Lorsque vous transmettez une occurrence de l'un de ces objets à l'aide de la méthode `Worker.setSharedProperty()` ou de la méthode `MessageChannel.send()`, chaque programme de travail possède une référence au même objet sous-jacent. Les modifications apportées à une occurrence dans un programme de travail sont immédiatement disponibles dans les autres programmes de travail. En outre, si vous transmettez plusieurs fois la même occurrence de l'un de ces objets à un programme de travail, le moteur d'exécution ne crée par une nouvelle copie de l'objet dans le programme de travail de réception. Au lieu de cela, la même référence est réutilisée.

Autres techniques de partage de données

Outre les mécanismes de transmission de données propres aux programmes de travail, les programmes de travail peuvent également échanger des données à l'aide de l'une des API existantes prenant en charge le partage de données entre deux applications swf :

- objets partagés locaux
- écriture de données sur un fichier dans un programme de travail et lecture du fichier dans un autre programme de travail
- stockage et lecture de données à partir d'une base de données SQLite

Lorsque vous partagez une ressource entre deux ou plusieurs programmes de travail, vous devez généralement éviter que divers programmes de travail accèdent en même temps à cette ressource. Par exemple, si plusieurs programmes de travail accèdent à un fichier sur le système de fichiers local, vous risquez de perdre ou d'endommager les données ; en outre, le système d'exploitation risque de ne pas prendre en charge cette opération.

Pour éviter les problèmes liés aux accès simultanés, utilisez les classes Mutex et Condition du package flash.concurrent pour contrôler l'accès aux ressources partagées.

Contrairement à d'autres mécanismes de partage de données, le moteur de la base de données SQLite est conçu pour un accès simultané et possède un processus transactionnel propre. Plusieurs programmes de travail peuvent accéder à une base de données SQLite sans risquer d'endommager les données. Etant donné que les programmes de travail utilisent différentes occurrences de SQLConnection, chaque programme de travail accède à la base de données dans une transaction distincte. Les opérations de manipulation simultanée de données n'ont aucune incidence sur l'intégrité des données.

Voir aussi

« Utilisation des bases de données SQL locales dans AIR » à la page 741
Package flash.concurrent

Chapitre 64 : Sécurité

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La sécurité est une préoccupation essentielle pour Adobe, les utilisateurs, les propriétaires de sites Web et les développeurs de contenu. C'est la raison pour laquelle Adobe® Flash® Player et Adobe® AIR™ incluent un ensemble de contrôles et de règles de sécurité qui protègent l'utilisateur, le propriétaire du site Web et le développeur du contenu. Les sections ci-après passent en revue le modèle de sécurité des fichiers SWF publiés en ActionScript 3.0 et qui s'exécutent dans Flash Player 9.0.124.0 ou ultérieur, ainsi que des fichiers SWF, HTML et JavaScript qui s'exécutent dans AIR 1.0 ou ultérieur, sauf mention contraire.

Les présentes sections visent à offrir une présentation générale de la sécurité et non à apporter une explication exhaustive de tous les détails de l'implémentation, des scénarios d'exploitation ou des ramifications de l'utilisation de certaines API. Pour plus d'informations sur les concepts de sécurité de Flash Player, voir la rubrique « Sécurité » du Centre des développeurs de Flash Player à l'adresse www.adobe.com/go/devnet_security_fr.

Présentation de la sécurité sur la plate-forme Flash

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le modèle de sécurité utilisé par les moteurs d'exécution de Flash Player et d'AIR est en grande partie basé sur le domaine d'origine des fichiers SWF, des contenus HTML, des éléments multimédias et autres actifs chargés. Le code exécutable d'un fichier issu d'un domaine Internet particulier, tel www.exemple.com, peut systématiquement accéder à l'ensemble des données de ce domaine. Ces actifs sont placés dans le même groupe de sécurité, appelé *sandbox de sécurité* (pour plus d'informations, voir « Sandbox de sécurité » à la page 1087).

Par exemple, le code ActionScript d'un fichier SWF peut charger des fichiers SWF, bitmap, audio, texte et tout autre actif appartenant au même domaine. En outre, l'intercodage entre deux fichiers SWF d'un même domaine est systématiquement autorisé, sous réserve qu'ils soient tous deux écrits en ActionScript 3.0. L'*intercodage* est la capacité du code d'un fichier à accéder aux propriétés, méthodes et objets définis par le code d'un autre fichier.

Elle n'est pas prise en charge si les fichiers sont écrits en ActionScript 3.0 pour certains et dans des versions antérieures d'ActionScript pour d'autres. La communication entre ces fichiers sera toutefois possible grâce à la classe LocalConnection. Par ailleurs, un fichier SWF ne peut pas, par défaut, accéder par programmation croisée à des fichiers SWF écrits en ActionScript 3.0 appartenant à d'autres domaines ni charger des données à partir d'autres domaines. Ce type d'accès est néanmoins autorisé par le biais d'un appel à la méthode `Security.allowDomain()` dans le fichier SWF chargé. Pour plus de détails sur la programmation croisée, voir « Programmation croisée » à la page 1107.

Les règles de sécurité élémentaires présentées ci-après s'appliquent toujours par défaut :

- Des ressources issues du même sandbox de sécurité ont accès les unes aux autres.
- Le code exécutable des fichiers situés dans un sandbox distant ne peut jamais accéder aux données et fichiers locaux.

Les moteurs d'exécution de Flash Player et d'AIR considèrent les éléments suivants comme des domaines distincts et configurent un sandbox de sécurité pour chacun d'eux :

- `http://example.com`
- `http://www.example.com`

- `http://store.example.com`
- `https://www.example.com`
- `http://192.0.34.166`

Même si un domaine nommé, tel http://example.com, est associé à une adresse IP particulière, telle http://192.0.34.166, les moteurs d'exécution définissent un sandbox distinct pour chacun d'eux.

Le développeur dispose de deux méthodes de base pour permettre à un fichier SWF d'accéder à des actifs issus de sandbox différents du sien.

- La méthode `Security.allowDomain()` (voir « Contrôles de création (développeur) » à la page 1099)
- Le fichier de régulation d'URL (voir « Contrôles de site Web (fichiers de régulation) » à la page 1095)

Les modèles de sécurité utilisés par les moteurs d'exécution de Flash Player et d'AIR établissent une distinction entre le chargement de contenu et l'accès à des données ou leur extraction. On entend par *contenu* des fichiers multimédias, notamment les éléments visuels que les moteurs d'exécution peuvent afficher, des fichiers audio et vidéo ou un fichier SWF ou HTML contenant des éléments multimédias affichés. Les *données* sont des éléments auxquels seul le code peut accéder. Le contenu et les données ne sont pas chargés de la même façon.

- Chargement de contenu : vous pouvez charger un contenu par le biais de classes telles que Loader, Sound et NetStream, à l'aide de balises MXML si vous utilisez Flex ou à l'aide de balises HTML dans une application AIR.
- Extraction de données : vous pouvez extraire des données de contenus multimédias chargés à l'aide d'objets Bitmap, des méthodes `BitmapData.draw()` et `BitmapData.drawWithQuality()`, de la propriété `Sound.id3` ou de la méthode `SoundMixer.computeSpectrum()` La méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures.
- Accès aux données : vous pouvez accéder directement aux données en les chargeant à partir d'un fichier externe (un fichier XML, par exemple) par le biais de classes telles que URLStream, URLLoader, FileReference, Socket et XMLSocket. AIR propose d'autres classes de chargement de données, telles que FileStream et XMLHttpRequest.

Le modèle de sécurité Flash Player définit différentes règles concernant le chargement de contenu et l'accès aux données. En général, les restrictions sont moindres sur le chargement de contenu que sur l'accès aux données.

En règle générale, le contenu (fichiers SWF, bitmap, mp3 et vidéo) peut être chargé de n'importe quelle source, mais s'il provient d'un domaine autre que celui du code ou du contenu à l'origine du chargement, il est placé dans un sandbox de sécurité distinct.

Certains obstacles s'appliquent au chargement de contenu :

- Par défaut, les fichiers SWF locaux (ceux chargés à partir d'une adresse ne se trouvant pas sur un réseau, par exemple le disque dur de l'utilisateur) sont classés dans le sandbox local avec système de fichiers. Ces fichiers ne peuvent pas charger de contenu provenant d'un réseau. Pour plus d'informations, voir « Sandbox locaux » à la page 1087.
- Les serveurs RTMP (Real-Time Messaging Protocol) peuvent limiter l'accès au contenu. Pour plus d'informations, voir « Contenu diffusé à l'aide de serveurs RTMP » à la page 1107.

Si le média chargé est une image, un fichier audio ou une vidéo, ses données (par exemple données de pixels ou sons) sont accessibles pour un fichier SWF situé en dehors de son sandbox de sécurité, à condition que le domaine de ce fichier SWF ait été inclus dans un fichier de régulation d'URL dans le domaine d'origine du média. Pour plus d'informations, voir « Accès aux médias chargés comme s'il s'agissait de données » à la page 1111.

Les données chargées peuvent également provenir de fichiers texte ou XML chargés avec l'objet URLLoader. Là encore, l'accès à des données situées dans un autre sandbox de sécurité nécessite l'attribution des autorisations suffisantes par le biais d'un fichier de régulation d'URL placé dans le domaine d'origine. Pour plus d'informations, voir « Utilisation de URLLoader et URLStream » à la page 1113.

Remarque : *il n'est jamais nécessaire de faire appel à des fichiers de régulation pour permettre au code qui s'exécute dans le sandbox de l'application AIR de charger du contenu ou des données distants.*

Sandbox de sécurité

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les ordinateurs clients peuvent obtenir des fichiers individuels contenant du code, un contenu et des données à partir de sources diverses, telles que des sites Web externes, un système de fichiers local ou une application AIR installée. Les moteurs d'exécution de Flash Player et d'AIR associent chaque fichier de code et autres ressources (objets partagés, fichiers bitmap, son, vidéo et données) à des sandbox de sécurité en fonction de leur origine au moment du chargement. Les sections suivantes passent en revue les règles mises en place par les moteurs d'exécution pour contrôler ce à quoi un code ou un contenu peut accéder au sein d'un sandbox donné.

Pour plus de détails sur la sécurité de Flash Player, voir la rubrique « Sécurité » du Centre des développeurs Flash Player à l'adresse www.adobe.com/go/devnet_security_fr.

Sandbox distants

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les moteurs d'exécution de Flash Player et d'AIR classent les actifs (notamment les fichiers SWF) issus d'Internet dans des sandbox distincts correspondant à leur domaine d'origine. Les actifs chargés d'*example.com* ne partagent par exemple pas le même sandbox de sécurité que les actifs issus de *foo.org*. Par défaut, ces fichiers sont autorisés à accéder à toutes les ressources issues de leur propre serveur. Il est possible d'autoriser les fichiers SWF distants à accéder à des données d'autres domaines à l'aide d'autorisations explicites portant sur les sites Web et les auteurs, par exemple des fichiers de régulation d'URL et la méthode `Security.allowDomain()`. Pour plus d'informations, voir « Contrôles de site Web (fichiers de régulation) » à la page 1095 et « Contrôles de création (développeur) » à la page 1099.

Les fichiers SWF distants ne peuvent pas charger de fichiers ou de ressources locales.

Pour plus de détails sur la sécurité de Flash Player, voir la rubrique « Sécurité » du Centre des développeurs Flash Player à l'adresse www.adobe.com/go/devnet_security_fr.

Sandbox locaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un fichier est dit *local* s'il est référencé par le biais du protocole `file:` ou d'un chemin UNC (Universal Naming Convention). Les fichiers SWF locaux sont placés dans l'un de quatre sandbox locaux :

- Sandbox local avec système de fichiers : pour raisons de sécurité, les moteurs d'exécution de Flash Player et AIR placent par défaut tous les fichiers locaux dans le sandbox local avec système de fichiers. A partir de ce sandbox, le code exécutable peut lire les fichiers locaux (à l'aide de la classe URLLoader, par exemple), mais en aucun cas communiquer avec le réseau. Ceci garantit à l'utilisateur que les données locales ne peuvent pas filtrer hors du réseau ou autrement être partagées de manière inopportune.

- Sandbox local avec réseau : lors de la compilation d'un fichier SWF, vous pouvez spécifier s'il dispose d'un accès réseau lorsqu'il est exécuté comme fichier local (voir « Définition du type de sandbox des fichiers SWF locaux » à la page 1090). De tels fichiers sont placés dans le sandbox local avec réseau. Les fichiers SWF placés dans le sandbox local avec réseau abandonnent leur accès aux fichiers locaux. En échange, ils sont autorisés à accéder aux données sur le réseau. Toutefois, un fichier local avec réseau ne peut pas lire des données dérivées du réseau si aucune autorisation n'est accordée pour cela par le biais d'un fichier de régulation d'URL ou d'un appel à la méthode`Security.allowDomain()`. A cet effet, le fichier de régulation d'URL doit accorder une autorisation à *tous* les domaines en utilisant `allow-access-from domain="*"/` ou `Security.allowDomain("*")`. Pour plus d'informations, voir « Contrôles de site Web (fichiers de régulation) » à la page 1095 et « Contrôles de création (développeur) » à la page 1099.
- Sandbox local approuvé : les fichiers SWF locaux enregistrés comme approuvés (par l'utilisateur ou un programme d'installation) sont placés dans ce sandbox. Les administrateurs système et les utilisateurs peuvent aussi associer un fichier SWF local au sandbox local approuvé ou l'en dissocier, selon les contraintes de sécurité (voir « Contrôles administrateur » à la page 1092 et « Contrôles utilisateur » à la page 1093). Les fichiers SWF associés au Sandbox local approuvé peuvent interagir avec tous les autres fichiers SWF et charger des données à partir de n'importe quel emplacement (à distance ou localement).
- Sandbox de sécurité de l'application AIR : ce sandbox contient du contenu installé à l'aide de l'application AIR en cours d'exécution. Par défaut, le code exécuté dans le sandbox de l'application AIR peut accéder par intercodage au code de n'importe quel domaine. En revanche, les fichiers qui résident hors du sandbox de l'application AIR ne peuvent pas accéder par intercodage au code de ce sandbox. Par défaut, le code et le contenu du sandbox de sécurité de l'application AIR peuvent charger le contenu et les données de n'importe quel domaine.

La communication entre le sandbox local avec réseau et le sandbox local avec système de fichiers est strictement interdite, tout comme la communication entre le sandbox local avec système de fichiers et le sandbox distant. Elles ne peuvent pas être autorisées par une application Flash Player ni par un utilisateur ou un administrateur.

La programmation croisée entre les fichiers HTML et SWF locaux (par exemple à l'aide de la classe ExternalInterface) exige que les deux fichiers impliqués se trouvent dans le sandbox local approuvé. Cette contrainte vient du fait que les modèles de sécurité locaux des navigateurs diffèrent de celui de Flash Player.

Les fichiers SWF du sandbox local avec réseau ne peuvent pas charger des fichiers SWF du sandbox local avec système de fichiers. Les fichiers SWF du sandbox local avec système de fichier ne peuvent pas charger des fichiers SWF du sandbox local avec réseau.

Sandbox de l'application AIR

Adobe AIR 1.0 et les versions ultérieures

Le moteur d'exécution d'Adobe AIR ajoute un autre sandbox, dit sandbox de l'*application*, au modèle de sandbox de sécurité de Flash Player. Les fichiers installés dans le cadre d'une application AIR sont chargés dans le sandbox de l'application. A tout autre fichier chargé par l'application s'appliquent des restrictions de sécurité correspondant aux limites stipulées par le modèle de sécurité de Flash Player standard.

Lors de l'installation d'une application, tous les fichiers intégrés à un package AIR sont installés sur l'ordinateur de l'utilisateur dans le répertoire de l'application. Les développeurs peuvent référencer ce répertoire dans le code via le modèle d'URL `app:/` (voir « Modèles d'URI » à la page 845). Tous les fichiers qui se trouvent dans l'arborescence du répertoire de l'application sont affectés au sandbox de l'application lorsque celle-ci est exécutée. Le contenu du sandbox de l'application est doté de tous les privilèges disponibles pour une application AIR, y compris l'interaction avec le système des fichiers locaux.

De nombreuses applications AIR n'utilisent que ces fichiers installés localement pour exécuter l'application. Toutefois, les applications AIR ne se limitent pas uniquement aux fichiers contenus dans le répertoire de l'application ; elles peuvent charger tout type de fichier à partir de n'importe quelle source. Ceci inclut aussi bien les fichiers locaux par rapport à l'ordinateur de l'utilisateur que ceux accessibles à partir des sources externes, comme ceux sur un réseau local ou sur Internet. Le type de fichier n'a pas d'impact sur les restrictions relatives à la sécurité ; les fichiers HTML chargés sont dotés des mêmes privilèges de sécurité que les fichiers SWF chargés à partir de la même source.

Le contenu du sandbox de sécurité de l'application a accès aux interfaces de programmation d'AIR que le contenu des autres sandbox n'est pas autorisé à utiliser. Par exemple, la propriété `air.NativeApplication.nativeApplication.applicationDescriptor`, qui renvoie les contenus du fichier descripteur d'application pour l'application, est limitée au contenu du sandbox de sécurité de l'application. Un autre exemple d'interface de programmation limitée : la classe FileStream qui contient des méthodes pour la lecture et l'écriture dans le système de fichiers local.

Les interfaces de programmation ActionScript réservées au contenu du sandbox de sécurité de l'application sont dotées du logo AIR dans le manuel *Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash*. L'utilisation de ces interfaces de programmation dans d'autres sandbox provoque le renvoi d'une exception SecurityError.

Pour un contenu HTML, dans un objet HTMLLoader, toutes les interfaces de programmation JavaScript d'AIR (celles qui sont disponibles via la propriété `window.runtime` ou l'objet `air` lors de l'utilisation du fichier AIRAliases.js), sont mises à la disposition du contenu dans le sandbox de sécurité de l'application. Le contenu HTML d'un autre sandbox n'ayant pas accès à la propriété `window.runtime`, il lui est impossible d'accéder aux interfaces de programmation AIR ou Flash Player.

Les restrictions complémentaires suivantes s'appliquent au content exécuté au sein du sandbox de l'application AIR :

- Pour un contenu HTML dans le sandbox de sécurité de l'application, il existe des limitations dans l'utilisation des interfaces de programmation qui peuvent transformer dynamiquement des chaînes en code exécutable après le chargement du code. Ceci empêche l'application d'injecter (et d'exécuter) du code par inadvertance à partir de sources non-applicatives, comme des domaines du réseau potentiellement non sécurisés. L'utilisation de la fonction `eval()` en est un exemple. Pour plus d'informations, voir « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130.
- Pour prévenir les attaques éventuelles d'hameçonnage, les balises `img` du contenu HTML des objets TextField d'ActionScript sont ignorées dans un contenu SWF au sein du sandbox de sécurité de l'application.
- Le contenu du sandbox de l'application ne peut pas utiliser le protocole `asfunction` du contenu HTML dans les champs de texte ActionScript 2.0.
- Le contenu SWF du sandbox de l'application ne peut pas utiliser la mémoire cache interdomaines, une fonction qui a été ajoutée à Flash Player 9 Mise à jour 3. Cette fonction permet à Flash Player de mettre constamment en mémoire cache le contenu du composant de la plate-forme Adobe et de le réutiliser à la demande dans du contenu SWF chargé. Cela lui évite de recharger le contenu plusieurs fois.

Restrictions associées au contenu JavaScript dans AIR

Adobe AIR 1.0 et les versions ultérieures

Contrairement au contenu du sandbox de sécurité de l'application, celui de JavaScript dans un sandbox de sécurité hors application *peut* à tout moment appeler la fonction `eval()` pour exécuter dynamiquement le code généré. Des restrictions s'appliquent néanmoins dans AIR au contenu JavaScript exécuté dans un sandbox de sécurité hors application, Parmi ces API figurent :

- Le code JavaScript dans un sandbox hors application n'a pas accès à l'objet `window.runtime` et, de ce fait, il ne peut pas exécuter d'interfaces de programmation AIR.

- Par défaut, le contenu dans un sandbox de sécurité hors application ne peut pas utiliser d'appels XMLHttpRequest pour charger des données à partir d'autres domaines ; il est limité au domaine d'où l'appel est lancé. Toutefois, le code de l'application peut autoriser le contenu hors application à le faire en paramétrant l'attribut `allowCrossdomainXHR` dans l'image conteneur ou l'iframe. Pour plus d'informations, voir « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130.
- Il existe des restrictions relatives aux appels de la méthode `window.open` de JavaScript. Pour plus de détails, voir la section « Restrictions relatives à l'appel de la méthode window.open() de JavaScript » à la page 1133.
- Un contenu HTML dans un sandbox de sécurité distant (réseau) ne peut charger que du contenu CSS, `image`, `iframe` et `img` à partir d'un domaine distant (à partir d'URL du réseau).
- Un contenu HTML dans un sandbox local avec système de fichiers, local avec réseau ou approuvé localement ne peut charger que du contenu CSS, `image`, `iframe` et `img` à partir de sandbox locaux (mais pas à partir d'URL d'application ou de réseau).

Pour plus d'informations, voir « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130.

Définition du type de sandbox des fichiers SWF locaux

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un utilisateur ou l'administrateur d'un ordinateur peut spécifier si un fichier SWF local est approuvé, lui permettant ainsi de charger des données de tous les domaines, locaux ou réseau. Cette caractéristique est définie dans les répertoires Flash Player Trust global et utilisateur. Pour plus d'informations, voir « Contrôles administrateur » à la page 1092 et « Contrôles utilisateur » à la page 1093.

Pour plus d'informations sur les sandbox locaux, voir « Sandbox locaux » à la page 1087.

Adobe Flash Professional

Vous pouvez configurer un fichier SWF pour le sandbox local avec système de fichiers ou le sandbox local avec réseau en définissant les paramètres de publication du document dans l'outil de création.

Adobe Flex

Vous pouvez configurer un fichier SWF pour le sandbox local avec système de fichiers ou le sandbox local avec réseau en définissant l'indicateur `use-network` dans le compilateur Adobe Flex. Pour plus d'informations, voir « A propos des options du compilateur d'applications » dans *Développement et déploiement d'applications Adobe Flex 3*.

Propriété Security.sandboxType

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La propriété statique en lecture seule `Security.sandboxType` permet au créateur d'un fichier SWF de déterminer le type de sandbox auquel le moteur d'exécution de Flash Player ou d'AIR a associé le fichier SWF. La classe Security inclut des constantes qui représentent les valeurs possibles de la propriété `Security.sandboxType`, comme suit :

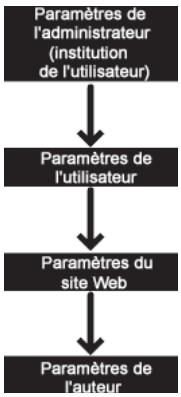
- `Security.REMOTE` : le fichier SWF provient d'une URL Internet et son fonctionnement est régi par les règles de sandbox de domaines.

- `Security.LOCAL_WITH_FILE` : le fichier SWF est un fichier local mais n'a pas été approuvé par l'utilisateur ni publié avec une désignation réseau. Le fichier SWF peut lire les sources de données locales mais ne peut pas communiquer avec Internet.
- `Security.LOCAL_WITH_NETWORK` : le fichier SWF est un fichier local non approuvé par l'utilisateur mais qui a été publié avec une désignation réseau. Le fichier SWF peut communiquer sur Internet mais ne peut pas lire les sources de données locales.
- `Security.LOCAL_TRUSTED` : le fichier SWF est un fichier local qui a été approuvé par l'utilisateur via le Gestionnaire des paramètres ou le fichier de configuration Flash Player Trust. Ce fichier SWF peut lire les sources de données locales et communiquer avec Internet.
- `Security.APPLICATION` : le fichier SWF est exécuté dans une application AIR, et a été installé avec le package (le fichier AIR) pour cette application. Par défaut, les fichiers du sandbox de sécurité de l'application AIR peuvent accéder par programmation croisée aux fichiers de n'importe quel domaine. En revanche, les fichiers se trouvant en dehors de ce sandbox ne peuvent pas accéder par programmation croisée au fichier AIR. Par défaut, les fichiers du sandbox de sécurité de l'application AIR peuvent charger le contenu et les données de n'importe quel domaine.

Contrôles des autorisations

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le modèle de sécurité d'exécution du client Flash Player repose sur les ressources, c'est-à-dire des objets tels que des fichiers SWF, des données locales et des URL Internet. Les *parties prenantes* détiennent ou utilisent ces ressources. Elles peuvent exercer un contrôle (via des paramètres de sécurité) sur leurs propres ressources, chaque ressource ayant quatre parties prenantes. Flash Player applique une stricte hiérarchie d'autorité sur ces contrôles, comme le montre l'illustration suivante :



Hiérarchie des contrôles de sécurité

Ainsi, par exemple, si un administrateur restreint l'accès à une ressource, aucune autre partie prenante ne peut revenir sur cette restriction.

Pour une application AIR, ces contrôles d'autorisations ne s'appliquent qu'au contenu exécuté en dehors de son sandbox.

Contrôles administrateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'administrateur d'un ordinateur (un utilisateur ayant ouvert une session avec des droits d'administration) peut définir des paramètres de sécurité Flash Player qui s'appliquent à tous les utilisateurs de l'ordinateur. Dans un environnement autre que l'entreprise, par exemple un ordinateur domestique, un utilisateur dispose aussi d'un accès administrateur. Même au sein d'une entreprise, certains utilisateurs peuvent posséder des droits d'administration sur l'ordinateur.

Il existe deux types de contrôles administrateur :

- Fichier mms.cfg
- Répertoire Flash Player Trust global

Fichier mms.cfg

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le fichier mms.cfg est un fichier texte qui permet à l'administrateur d'autoriser ou de limiter l'accès à diverses fonctionnalités. Lors du lancement de Flash Player, l'application lit les paramètres de sécurité dans ce fichier, puis les utilise pour limiter la fonctionnalité. Le fichier mms.cfg contient des paramètres utilisés par l'administrateur pour gérer des fonctionnalités telles que les contrôles de confidentialité, la sécurité des fichiers locaux, les connexions socket, etc.

Un fichier SWF peut obtenir des informations sur les fonctions désactivées en appelant les propriétés `Capabilities.avHardwareDisable` et `Capabilities.localFileReadDisable`. Toutefois, la plupart des paramètres du fichier mms.cfg ne peuvent être interrogés à partir d'ActionScript.

Pour mettre en place sur un ordinateur des stratégies de confidentialité et de sécurité indépendantes des applications, le fichier mms.cfg doit être uniquement modifié par un administrateur système. Le fichier mms.cfg n'est pas destiné aux programmes d'installation des applications. Bien qu'un programme d'installation exécuté avec des privilèges administrateur puisse modifier le contenu du fichier mms.cfg, Adobe considère cette pratique comme une violation de la confiance de l'utilisateur et presse les créateurs de programmes d'installation de ne jamais modifier le fichier mms.cfg.

Le fichier mms.cfg est enregistré dans le chemin suivant :

- Windows : *system*\Macromed\Flash\mms.cfg

  (par exemple, C:\windows\system32\Macromed\Flash\mms.cfg)

- Mac : *app support*/Macromedia/mms.cfg

  (par exemple, /Bibliothèque/Application Support/Macromedia/mms.cfg)

Pour plus d'informations sur le fichier mms.cfg, voir le Guide d'administration de Flash Player à l'adresse suivante : www.adobe.com/go/flash_player_admin_fr.

Répertoire Flash Player Trust global

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les administrateurs et les programmes d'installation peuvent approuver des fichiers SWF locaux spécifiques pour tous les utilisateurs. Ces fichiers SWF sont associés au sandbox local approuvé. Ils peuvent interagir avec tout autre fichier SWF puisqu'ils peuvent charger des données stockées localement ou à distance. Les fichiers approuvés dans le répertoire Flash Player Trust global se trouvent dans le chemin suivant :

- Windows : *system*\Macromed\Flash\FlashPlayerTrust

  (par exemple, C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac : *app support*/Macromedia/FlashPlayerTrust

  (par exemple, /Bibliothèque/Application Support/Macromedia/FlashPlayerTrust)

Le répertoire Flash Player Trust peut contenir un nombre illimité de fichiers texte, chacun répertoriant des chemins d'accès approuvés, à raison d'un chemin par ligne. Chaque chemin d'accès peut mener à un fichier SWF ou HTML, ou à un répertoire. Les lignes de commentaire commencent par le symbole #. Par exemple, un fichier de configuration Flash Player Trust contenant le texte suivant approuve tous les fichiers placés dans le répertoire spécifié et ses sous-répertoires :

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

Les chemins répertoriés dans un fichier de configuration Trust doivent toujours faire référence au système local ou à un réseau SMB. Les chemins d'accès HTTP placés dans un fichier de configuration Trust sont ignorés; seuls les fichiers locaux peuvent être approuvés.

Pour éviter les conflits, attribuez au fichier de configuration Trust un nom correspondant à l'application installée, suivi de l'extension de fichier .cfg.

En tant que développeur diffusant un fichier SWF à exécuter localement par le biais d'un programme d'installation, vous pouvez faire en sorte que ce programme d'installation ajoute un fichier de configuration au répertoire Flash Player Trust global afin d'accorder des privilèges complets au fichier que vous diffusez. Le programme d'installation doit être exécuté par un utilisateur doté de droits d'administration. Contrairement au fichier mms.cfg, le répertoire Flash Player Trust global est prévu pour les programmes d'installation devant accorder des autorisations. Il permet aux administrateurs et aux programmes d'installation de désigner des applications locales approuvées.

Il existe également des répertoires Flash Player Trust destinés aux utilisateurs individuels (voir « Contrôles utilisateur » à la page 1093).

Contrôles utilisateur

Flash Player 9 et les versions ultérieures

Flash Player propose trois mécanismes de définition des autorisations au niveau utilisateur : l'interface de paramétrage, le Gestionnaire des paramètres et le répertoire Flash Player Trust utilisateur.

Interface de paramétrage et Gestionnaire des paramètres

Flash Player 9 et les versions ultérieures

L'interface de paramétrage est un mécanisme interactif qui permet de configurer rapidement les paramètres d'un domaine donné. Le Gestionnaire des paramètres, avec son interface plus détaillée, permet d'effectuer des modifications globales sur les autorisations de plusieurs domaines ou de tous. En outre, si un fichier SWF nécessite une nouvelle autorisation qui oblige à prendre des décisions en cours d'exécution concernant la sécurité ou la confidentialité, des boîtes de dialogue s'affichent dans lesquelles les utilisateurs peuvent régler certains paramètres de Flash Player.

Le Gestionnaire des paramètres et l'interface de paramétrage proposent des options associées à la sécurité telles que les paramètres de la caméra et du microphone, les paramètres de stockage d'objets partagés, les paramètres relatifs au contenu hérité, etc. Les applications AIR ne disposent ni du Gestionnaire des paramètres, ni de l'interface de paramétrage.

Remarque : *les réglages effectués dans le fichier mms.cfg (voir « Contrôles administrateur » à la page 1092) n'apparaissent pas dans le Gestionnaire des paramètres.*

Pour plus d'informations sur le Gestionnaire des paramètres, voir www.adobe.com/go/settingsmanager_fr.

Répertoire Flash Player Trust utilisateur

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les utilisateurs et programmes d'installation peuvent approuver des fichiers SWF locaux spécifiques. Ces fichiers SWF sont associés au sandbox local approuvé. Ils peuvent interagir avec tout autre fichier SWF puisqu'ils peuvent charger des données stockées localement ou à distance. Pour approuver un fichier, l'utilisateur le désigne dans le répertoire Player Trust utilisateur, qui se trouve dans le même répertoire que la zone de stockage des objets partagés Flash, aux emplacements suivants (ces emplacements sont spécifiques à l'utilisateur actif) :

- Windows : app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (par exemple, C:\Documents and Settings\JohnD\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust sous Windows XP ou C:\Users\JohnD\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust sous Windows Vista)

  Sous Windows, le dossier Application Data est caché par défaut. Pour afficher les dossiers et fichiers cachés, sélectionnez le Poste de travail pour ouvrir l'Explorateur Windows, choisissez Outils > Options des dossiers et sélectionnez l'onglet Affichage. Sous l'onglet Affichage, sélectionnez le bouton d'option Afficher les fichiers et les dossiers cachés.

- Mac : app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (par exemple, /Utilisateurs/JohnD/Bibliothèque/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  Ces paramètres s'appliquent uniquement à l'utilisateur actif, et non aux autres utilisateurs qui ouvrent une session sur l'ordinateur. Si un utilisateur sans droits d'administration installe une application dans sa partie réservée du système, le répertoire Flash Player Trust utilisateur permet au programme d'installation d'enregistrer cette application comme approuvée pour l'utilisateur en question.

En tant que développeur diffusant un fichier SWF à exécuter localement par le biais d'un programme d'installation, vous pouvez faire en sorte que ce programme d'installation ajoute un fichier de configuration au répertoire Flash Player Trust utilisateur afin d'accorder des privilèges complets au fichier que vous diffusez. Même dans ce cas de figure, le fichier placé dans le répertoire Flash Player Trust utilisateur constitue un contrôle utilisateur puisque son initialisation résulte d'une action de l'utilisateur (l'installation).

Il existe également un répertoire Flash Player Trust global utilisé par l'administrateur ou les programmes d'installation pour enregistrer une application destinée à l'ensemble des utilisateurs d'un ordinateur (voir « Contrôles administrateur » à la page 1092).

Contrôles de site Web (fichiers de régulation)

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si vous souhaitez rendre les données d'un serveur Web accessibles aux fichiers SWF issus de domaines différents, vous pouvez créer un fichier de régulation sur votre serveur. Un *fichier de régulation* est un fichier XML résidant à un emplacement spécifique sur votre serveur.

Les fichiers de régulation ont une incidence sur l'accès à certains actifs, notamment les suivants :

- Les données de fichiers bitmap, son et vidéo
- Le chargement des fichiers XML et texte
- L'importation de fichiers SWF à partir d'autres domaines de sécurité dans le domaine du fichier SWF à l'origine du chargement
- L'accès aux connexions socket et socket XML

Les objets ActionScript instancient deux types de connexions serveur : les connexions serveur liées à des documents et les connexions socket. Les objets ActionScript tels que Loader, Sound, URLLoader et URLStream instancient des connexions serveur liées à des documents et chargent un fichier à partir d'une URL. Les objets ActionScript Socket et XMLSocket établissent des connexions socket, qui fonctionnent avec des flux de données et non des documents chargés.

Flash Player prenant en charge deux types de connexions serveur, il existe deux types de fichiers de régulation : les fichiers de régulation d'URL et les fichiers de régulation socket.

- Les connexions liées à des documents exigent des *fichiers de régulation d'URL*. Ces fichiers permettent au serveur d'indiquer que ses données et documents sont accessibles aux fichiers SWF servis à partir de certains domaines ou de tous les domaines.
- Les connexions socket requièrent des *fichiers de régulation socket*, qui permettent un accès réseau direct au niveau socket TCP le plus bas, à l'aide des classes Socket et XMLSocket.

Flash Player exige que les fichiers de régulation soient transmis par le biais du protocole utilisé par la tentative de connexion. Par exemple, si vous placez un fichier de régulation sur un serveur HTTP, les fichiers SWF issus d'autres domaines sont autorisés à charger les données qu'il contient en tant que serveur HTTP. Cependant, si vous ne fournissez aucun fichier de régulation socket sur ce même serveur, vous empêchez les fichiers SWF issus d'autres domaines d'accéder au serveur au niveau socket. La méthode de récupération du fichier de régulation doit donc correspondre à la méthode de connexion.

L'utilisation et la syntaxe des fichiers de régulation, tels qu'ils s'appliquent aux fichiers SWF publiés pour Flash Player 10, sont décrites dans les paragraphes suivants (l'implémentation des fichiers de régulation est légèrement différente dans les versions antérieures de Flash Player, dont la sécurité a été renforcée ultérieurement). Pour plus d'informations sur les fichiers de régulation, voir le chapitre « Modifications du fichier de régulation dans Flash Player 9 » dans le Centre des développeurs Flash Player à l'adresse www.adobe.com/go/devnet_security_fr.

Le code exécuté dans le sandbox de l'application AIR ne requiert pas de fichier de régulation pour accéder aux données issues d'une URL ou d'un socket. Le code d'une application AIR exécuté dans un sandbox hors application nécessite en revanche un fichier de régulation.

Fichiers de régulation maître

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, Flash Player (et le contenu AIR qui ne figure pas dans le sandbox d'application AIR) commence par rechercher le fichier de régulation d'URL `crossdomain.xml` dans le répertoire racine du serveur, puis recherche un fichier de régulation socket sur le port 843. Tout fichier résidant à l'un de ces emplacements constitue le *fichier de régulation maître* (dans le cas des connexions socket, Flash Player recherche également un fichier de régulation socket sur le port utilisé par la connexion principale. Un fichier de régulation se trouvant sur ce port ne constitue cependant pas un fichier de régulation maître).

Outre la définition des autorisations d'accès, le fichier de régulation maître peut également contenir une instruction *meta-policy*. Une méta-régulation détermine les emplacements auxquels peuvent résider les fichiers de régulation. La méta-régulation par défaut des fichiers de régulation d'URL correspond à « master-only » ; autrement dit, /crossdomain.xml est le seul fichier de régulation autorisé sur le serveur. La méta-régulation par défaut des fichiers de régulation socket correspond à « all » ; autrement dit, tout socket figurant sur l'hôte peut servir un fichier de régulation de socket.

Remarque : *dans Flash Player 9 et les versions antérieures, la méta-régulation par défaut des fichiers de régulation d'URL correspondait à « all ». Tout répertoire peut donc contenir un fichier de régulation. Si vous avez déployé des applications qui chargent d'autres fichiers de régulation que le fichier /crossdomain.xml par défaut, et que ces applications sont maintenant susceptibles de s'exécuter dans Flash Player 10, vous (ou l'administrateur du serveur) devez modifier le fichier de régulation maître pour autoriser l'utilisation d'autres fichiers de régulation. Pour plus d'informations sur la définition d'une méta-régulation différente, voir le chapitre « Modifications du fichier de régulation dans Flash Player 9 » dans le Centre des développeurs Flash Player à l'adresse www.adobe.com/go/devnet_security_fr.*

Un fichier SWF peut effectuer une recherche sur un nom de fichier de régulation ou un emplacement différent en appelant la méthode `Security.loadPolicyFile()`. Cependant, si le fichier de régulation maître ne spécifie pas que l'emplacement cible peut servir des fichiers de régulation, l'appel à `loadPolicyFile()` est sans effet, même si un fichier de régulation se trouve à l'emplacement en question. Appelez `loadPolicyFile()` avant d'exécuter toute opération réseau requérant le fichier de régulation. Flash Player place automatiquement les requêtes réseau derrière les tentatives de requête de fichier de régulation correspondantes. Il est donc acceptable, par exemple, d'appeler `Security.loadPolicyFile()` immédiatement avant de lancer une opération réseau.

Lorsqu'il recherche un fichier de régulation maître,. Flash Player attend une réponse du serveur pendant trois secondes. En l'absence d'une réponse, l'application considère comme acquis qu'il n'existe pas de fichier de régulation maître. Toutefois, si aucune valeur de dépassement de délai par défaut est définie pour les appels à `loadPolicyFile()`, Flash Player suppose que le fichier appelé existe et attend aussi longtemps que nécessaire pour le charger. Pour avoir la certitude qu'un fichier de régulation maître est chargé, appelez-le donc explicitement par le biais de `loadPolicyFile()`.

Bien que la méthode s'appelle `Security.loadPolicyFile()`, aucun fichier de régulation n'est chargé tant qu'un appel réseau requérant un tel fichier n'a pas été émis. Les appels à `loadPolicyFile()` indiquent simplement à Flash Player où rechercher les fichiers de régulation, le cas échéant.

Aucune notification indiquant l'initiation ou la fin d'une requête de fichier de régulation n'est envoyée, car cela n'est pas nécessaire. Flash Player recherche les fichiers de régulation de manière asynchrone et attend automatiquement que ces recherches aient abouti pour établir des connexions.

Les sections suivantes s'appliquent uniquement aux fichiers de régulation d'URL. Pour plus d'informations sur les fichiers de régulation socket, voir « Connexion aux sockets » à la page 1114.

Portée des fichiers de régulation d'URL

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un fichier de régulation d'URL s'applique uniquement au répertoire dans lequel il est chargé et à ses sous-répertoires. Un fichier de régulation placé dans le répertoire racine s'applique à l'ensemble du serveur. En revanche, un fichier de régulation chargé d'un sous-répertoire quelconque s'applique uniquement à celui-ci et à ses sous-répertoires.

Un fichier de régulation contrôle uniquement l'accès au serveur sur lequel il réside. Par exemple, un fichier de régulation situé dans https://www.adobe.com:8080/crossdomain.xml ne s'applique qu'aux appels de chargement de données passés vers www.adobe.com sur HTTPS au port 8080.

Définition des autorisations d'accès dans un fichier de régulation d'URL

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un fichier de régulation contient une seule balise `cross-domain-policy`, qui contient elle-même aucune ou plusieurs balises `allow-access-from`. Chaque balise `allow-access-from` contient un attribut `domain` qui spécifie une adresse IP exacte, un domaine exact ou un domaine générique (tout domaine). Les domaines génériques sont indiqués de deux façons :

- Par un astérisque (*) seul, qui représente tous les domaines et toutes les adresses IP
- Par un astérisque suivi d'un suffixe, qui représente uniquement les domaines se terminant par ce suffixe

Les suffixes doivent commencer par un point. Cependant, les domaines génériques suivis de suffixes peuvent correspondre à des domaines qui sont composés uniquement du suffixe sans le point de séparation. xyz.com, par exemple, fait partie de *.xyz.com. L'utilisation de caractères génériques est interdite dans les spécifications de domaine IP.

L'exemple suivant présente un fichier de régulation d'URL qui autorise l'accès à des fichiers SWF issus des domaines *.example.com, www.friendOfExample.com et 192.0.34.166 :

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="*.example.com" />
    <allow-access-from domain="www.friendOfExample.com" />
    <allow-access-from domain="192.0.34.166" />
</cross-domain-policy>
```

Si vous spécifiez une adresse IP, seuls les fichiers SWF chargés depuis cette adresse IP à l'aide de la syntaxe IP (par exemple, http://65.57.83.12/flashmovie.swf) sont accessibles ; les fichiers chargés à l'aide d'une syntaxe domaine-nom sont inaccessibles Flash Player n'effectue pas de résolution DNS.

Vous pouvez autoriser l'accès à des documents provenant de tout autre domaine, comme indiqué dans l'exemple suivant :

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

Chaque balise `allow-access-from` peut présenter l'attribut facultatif `secure`, dont la valeur par défaut est `true`. Si votre fichier de régulation se trouve sur un serveur HTTPS et que vous voulez autoriser les fichiers SWF résidant sur un autre type de serveur à charger des données à partir du serveur HTTPS, vous pouvez définir l'attribut sur `false`.

La définition de l'attribut `secure` sur `false` risque de compromettre la sécurité fournie par le protocole HTTPS. Plus particulièrement, la définition de cet attribut sur `false` rend le contenu sécurisé vulnérable aux attaques d'espionnage. Adobe recommande vivement de ne pas définir l'attribut `secure` sur `false`.

Si les données à charger se trouvent sur un serveur HTTPS, alors que le fichier SWF à l'origine du chargement réside sur un serveur HTTP, Adobe recommande le transfert du fichier SWF sur un serveur HTTPS. Ce faisant, toutes les copies de vos données sécurisées sont protégées par HTTPS. Cependant si vous décidez que vous devez conserver le fichier SWF à l'origine du chargement sur un serveur HTTP, ajoutez l'attribut `secure="false"` à la balise `allow-access-from`, comme le montre le code suivant :

```
<allow-access-from domain="www.example.com" secure="false" />
```

Pour autoriser l'accès, vous pouvez aussi utiliser la balise `allow-http-request-headers-from`. Elle permet à un client hébergeant du contenu issu d'un autre domaine d'autorisation d'envoyer des en-têtes définis par l'utilisateur à votre domaine. La balise `<allow-access-from>` autorise d'autres domaines à récupérer des données de votre domaine, tandis que la balise `allow-http-request-headers-from` permet à d'autres domaines d'envoyer des données, sous forme d'en-têtes, à votre domaine. Dans l'exemple ci-dessous, n'importe quel domaine est autorisé à envoyer l'en-tête SOAPAction au domaine en cours :

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

Si l'instruction `allow-http-request-headers-from` figure dans le fichier de régulation maître, elle s'applique à tous les répertoires de l'hôte. Dans le cas contraire, elle s'applique uniquement au répertoire, et sous-répertoires correspondants, du fichier de régulation contenant l'instruction.

Préchargement des fichiers de régulation

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le chargement des données à partir d'un serveur ou la connexion à un socket sont des opérations asynchrones. Flash Player attend simplement la fin du téléchargement du fichier de régulation avant de lancer l'opération principale. En revanche, l'extraction de données de pixel d'images ou de données d'échantillonnage de sons est une opération synchrone. Le fichier de régulation doit donc être chargé pour que vous puissiez extraire les données. Lors du chargement du média, spécifiez que le fichier de régulation doit être recherché :

- Si vous utilisez la méthode `Loader.load()`, définissez la propriété `checkPolicyFile` du paramètre `context`, qui constitue un objet LoaderContext.
- Si vous incorporez une image dans un champ de texte à l'aide de la balise `<img>`, définissez l'attribut `checkPolicyFile` de la balise `<img>` sur `true`, comme dans l'exemple suivant :

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- Si vous utilisez la méthode `Sound.load()`, définissez la propriété `checkPolicyFile` du paramètre `context`, qui constitue un objet SoundLoaderContext.
- Si vous utilisez la classe NetStream, définissez la propriété `checkPolicyFile` de l'objet NetStream.

Lorsque vous définissez l'un de ces paramètres, Flash Player commence par vérifier si des fichiers de régulation ont déjà été téléchargés pour ce domaine. Il recherche ensuite le fichier de régulation à l'emplacement par défaut sur le serveur, puis les instructions `<allow-access-from>` et une méta-régulation. En dernier lieu, il contrôle tout appel en attente à la méthode `Security.loadPolicyFile()` pour vérifier s'il s'applique.

Contrôles de création (développeur)

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

L'API ActionScript principale utilisée dans l'attribution des privilèges de sécurité est la méthode `Security.allowDomain()`, qui accorde des droits aux fichiers SWF du domaine que vous spécifiez. Dans l'exemple suivant, un fichier SWF autorise l'accès des fichiers SWF servis à partir du domaine www.example.com :

```
Security.allowDomain("www.example.com")
```

Cette méthode autorise les opérations suivantes :

- Programmation croisée entre fichiers SWF (voir« Programmation croisée » à la page 1107)
- Accès à la liste d'affichage (voir « Parcours de la liste d'affichage » à la page 1110)
- Détection d'événements (voir « Sécurité des événements » à la page 1110)
- Accès total aux propriétés et méthodes de l'objet Stage (voir « Sécurité de la scène » à la page 1109)

La méthode `Security.allowDomain()` sert avant tout à permettre aux fichiers SWF situés dans un domaine externe d'effectuer une programmation croisée avec le fichier SWF qui appelle la méthode `Security.allowDomain()`. Pour plus de détails sur la programmation croisée, voir « Programmation croisée » à la page 1107.

La spécification de l'adresse IP en tant que paramètre de `Security.allowDomain()` n'autorise pas l'accès de toutes les parties provenant de l'adresse IP spécifiée. Au contraire, elle autorise l'accès d'une partie présentant une URL identique à l'adresse IP spécifiée, plutôt qu'un nom de domaine renvoyant à l'adresse IP. Par exemple, si le nom de domaine www.example.com renvoie à l'adresse IP 192.0.34.166, un appel à `Security.allowDomain("192.0.34.166")` ne donne pas accès à www.example.com.

Vous pouvez transmettre le caractère générique `"*"` à la méthode `Security.allowDomain()` pour permettre l'accès à partir de tous les domaines. Soyez prudent lorsque vous utilisez le caractère générique "*" car celui-ci autorise les fichiers SWF issus de `tous` les domaines à effectuer une programmation dans le fichier SWF appelant.

ActionScript inclut une seconde API d'autorisation appelée `Security.allowInsecureDomain()`. Cette méthode joue le même rôle que la méthode `Security.allowDomain()` à la différence suivante : lorsqu'elle est appelée à partir d'un fichier SWF servi par une connexion HTTPS, elle autorise l'accès à ce fichier appelant pour d'autres fichiers SWF servis à partir d'un protocole non sécurisé, tel HTTP. Cependant, il est déconseillé de permettre la programmation croisée entre des fichiers issus d'un protocole sécurisé et ceux provenant d'un protocole qui ne l'est pas. Cette pratique est susceptible de rendre le contenu vulnérable aux attaques d'espionnage. Ces attaques se déroulent comme suit, par exemple : puisque la méthode `Security.allowInsecureDomain()` permet aux fichiers SWF servis sur des connexions HTTP d'accéder aux données HTTPS sécurisées, un attaquant qui s'interposerait entre le serveur HTTP et les utilisateurs pourraient remplacer votre fichier SWF HTTP par un fichier de son cru afin d'accéder à vos données HTTPS.

Important : *le code exécuté dans le sandbox de l'application AIR n'est pas autorisé à appeler la méthode* `allowDomain()` *ou* `allowInsecureDomain()` *de la classe Security.*

Autre méthode importante liée à la sécurité, `Security.loadPolicyFile()` permet à Flash Player de rechercher le fichier de régulation à un autre emplacement. Pour plus d'informations, voir « Contrôles de site Web (fichiers de régulation) » à la page 1095.

Restriction des API de réseau

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les API de réseau peuvent faire l'objet de deux types de restriction : Pour empêcher les activités nuisibles, l'accès aux ports généralement réservés est bloqué. Vous ne pouvez pas passer outre ces blocages dans votre code. Pour contrôler l'accès au réseau par le biais d'autres ports d'un fichier SWF, vous pouvez utiliser le paramètre `allowNetworking`.

Ports bloqués

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

A l'instar des navigateurs, Flash Player et Adobe AIR imposent des restrictions sur l'accès HTTP à certains ports. Les requêtes HTTP sont interdites sur certains ports standard traditionnellement réservés aux types de serveurs autres que HTTP.

Toute API accédant à une URL réseau est soumise aux restrictions affectant ces ports. Les API qui appellent directement des sockets, telles que `Socket.connect()` et `XMLSocket.connect()`, ou les appels à `Security.loadPolicyFile()` dans lesquels un fichier de régulation socket est en cours de chargement font exception à cette règle. Les connexions socket sont autorisées ou refusées par le biais de fichiers de régulation socket sur le serveur cible.

Les API ActionScript 3.0 concernées par le blocage des ports sont les suivantes :

`FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `Loader.loadBytes()`, `navigateToURL()`, `NetConnection.call()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `sendToURL()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`

Le blocage des ports s'applique également à l'importation dans une bibliothèque partagée, à l'utilisation de la balise `<img>` dans les champs de texte et au chargement de fichiers SWF sur une page HTML à l'aide des balises `<object>` et `<embed>`.

Le blocage des ports s'applique également à l'utilisation de la balise `<img>` dans les champs de texte et au chargement de fichiers SWF sur une page HTML à l'aide des balises `<object>` et `<embed>`.

Les ports bloqués sont les suivants :

HTTP: 20 (ftp data), 21 (ftp control)

HTTP et FTP : 1 (tcpmux), 7 (echo), 9 (discard), 11 (systat), 13 (daytime), 15 (netstat), 17 (qotd), 19 (chargen), 22 (ssh), 23 (telnet), 25 (smtp), 37 (time), 42 (name), 43 (nicname), 53 (domain), 77 (priv-rjs), 79 (finger), 87 (ttylink), 95 (supdup), 101 (hostriame), 102 (iso-tsap), 103 (gppitnp), 104 (acr-nema), 109 (pop2), 110 (pop3), 111 (sunrpc), 113 (auth), 115 (sftp), 117 (uucp-path), 119 (nntp), 123 (ntp), 135 (loc-srv / epmap), 139 (netbios), 143 (imap2), 179 (bgp), 389 (ldap), 465 (smtp+ssl), 512 (print / exec), 513 (login), 514 (shell), 515 (printer), 526 (tempo), 530 (courier), 531 (chat), 532 (netnews), 540 (uucp), 556 (remotefs), 563 (nntp+ssl), 587 (smtp), 601 (syslog), 636 (ldap+ssl), 993 (ldap+ssl), 995 (pop3+ssl), 2049 (nfs), 4045 (lockd), 6000 (x11)

Utilisation du paramètre allowNetworking

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez contrôler l'accès d'un fichier SWF aux fonctionnalités réseau en définissant le paramètre `allowNetworking` dans les balises `<object>` et `<embed>` de la page HTML qui accueille le contenu SWF.

`allowNetworking` peut prendre les valeurs suivantes :

- `"all"` (par défaut) : toutes les API de réseau sont autorisées dans le fichier SWF.
- `"internal"` : le fichier SWF ne peut pas appeler les API de navigation et d'interaction avec le navigateur (présentées plus loin dans cette section). Il peut cependant appeler toutes les autres API de réseau.
- `"none"` : le fichier SWF ne peut pas appeler les API de navigation et d'interaction avec le navigateur (présentées plus loin dans cette section) ni utiliser les API de communication entre fichiers SWF (également décrites plus loin).

Le paramètre `allowNetworking` s'utilise surtout lorsque le fichier SWF et la page HTML qui l'accueille appartiennent à des domaines différents. Il est déconseillé d'utiliser la valeur `"internal"` ou `"none"` si le fichier SWF en cours de chargement appartient au même domaine que les pages HTML qui l'accueillent, car vous ne pouvez pas garantir qu'un fichier SWF est toujours chargé avec la page HTML prévue. Des personnes mal intentionnées pourraient charger un fichier SWF à partir de votre domaine sans les pages HTML correspondantes, auquel cas la restriction `allowNetworking` n'aurait pas l'effet escompté.

L'appel d'une API interdite renvoie une exception SecurityError.

Ajoutez le paramètre `allowNetworking` et définissez sa valeur dans les balises `<object>` et `<embed>` de la page HTML contenant une référence au fichier SWF, comme indiqué dans l'exemple suivant :

```
<object classic="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
    Code
base="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,124,
0"
    width="600" height="400" ID="test" align="middle">
<param name="allowNetworking" value="none" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowNetworking="none" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Une page HTML peut également utiliser un script pour générer des balises d'imbrication de fichiers SWF. Vous devez modifier le script de façon à insérer les paramètres `allowNetworking` corrects. Les pages HTML générées par Adobe Flash Professional et Adobe Flash Builder utilisent la fonction `AC_FL_RunContent()` pour incorporer des références aux fichiers SWF. Ajoutez les paramètres `allowNetworking` au script, comme indiqué dans l'exemple suivant :

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

Les API suivantes ne sont pas autorisées lorsque `allowNetworking` est défini sur `"internal"` :

```
navigateToURL(), fscommand(), ExternalInterface.call()
```

Outre les API recensées ci-dessus, les API ci-après sont également interdites lorsque `allowNetworking` est défini sur `"none"` :

```
sendToURL(), FileReference.download(), FileReference.upload(), Loader.load(),
LocalConnection.connect(), LocalConnection.send(), NetConnection.connect(), NetStream.play(),
Security.loadPolicyFile(), SharedObject.getLocal(), SharedObject.getRemote(), Socket.connect(),
Sound.load(), URLLoader.load(), URLStream.load(), XMLSocket.connect()
```

Même si le paramètre `allowNetworking` sélectionné permet au fichier SWF d'utiliser une API de réseau, d'autres restrictions peuvent survenir en fonction des limites fixées par le sandbox de sécurité (voir « Sandbox de sécurité » à la page 1087).

Si `allowNetworking` est défini sur `"none"`, vous ne pouvez pas référencer un média externe dans une balise `<img>` de la propriété `htmlText` d'un objet TextField (si vous le faites, une exception SecurityError est renvoyée).

Si `allowNetworking` est défini sur `"none"`, un symbole issu d'une bibliothèque partagée importée dans Flash Professional (pas ActionScript) est bloqué lors de l'exécution.

Sécurité du mode plein écran

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Player version 9.0.27.0 et les versions ultérieures prennent en charge le mode plein écran, dans lequel le contenu Flash Player peut remplir tout l'écran. Pour passer en mode plein écran, la propriété `displayState` de la scène est définie avec la constante `StageDisplayState.FULL_SCREEN`. Pour plus d'informations, voir « Utilisation du mode Plein écran » à la page 173.

L'exécution de fichiers SWF dans un sandbox distant impose la prise en considération de certains points de sécurité.

L'activation du mode plein écran s'effectue dans les balises `<object>` et `<embed>` de la page HTML qui contient la référence au fichier SWF. Pour ce faire, ajoutez le paramètre `allowFullScreen` et attribuez-lui la valeur `"true"` (la valeur par défaut est `"false"`), comme le montre l'exemple suivant :

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,
18,0"
    width="600" height="400" id="test" align="middle">
<param name="allowFullScreen" value="true" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowFullScreen="true" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Une page HTML peut également utiliser un script pour générer des balises d'imbrication de fichiers SWF. Vous devez modifier le script de manière qu'il insère les paramètres `allowFullScreen` appropriés. Les pages HTML générées par Flash Professional et Flash Builder utilisent la fonction `AC_FL_RunContent()` pour incorporer des références aux fichiers SWF, et vous devez ajouter les paramètres `allowFullScreen`, comme dans l'exemple suivant :

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

Le code ActionScript qui lance le mode plein écran peut être uniquement appelé en réponse à un événement souris ou clavier. S'il est appelé dans d'autres situations, Flash Player renvoie une exception.

Lorsque le contenu passe en mode plein écran, un message s'affiche pour indiquer à l'utilisateur comment quitter ce mode et revenir au mode normal. Le message s'affiche pendant quelques secondes, puis s'estompe.

Pour le contenu qui s'exécute dans un navigateur, l'utilisation du clavier est restreinte en mode plein écran. Dans Flash Player 9, seuls les raccourcis clavier qui réinitialisent le mode normal de l'application (tel appuyer sur la touche Echap) sont pris en charge. Les utilisateurs ne sont pas autorisés à entrer du texte dans un champ de texte ou à naviguer à l'écran. Flash Player 10 (et versions ultérieures) prend en charge certaines touches hors impression, notamment les touches fléchées, la barre d'espacement et la touche de tabulation. La saisie de texte est néanmoins toujours interdite.

Le mode plein écran est toujours autorisé dans le lecteur autonome ou dans un fichier de projection. Par ailleurs, l'utilisation du clavier (y compris la saisie de texte) est totalement prise en charge dans ces environnements.

L'appel de la propriété `displayState` d'un objet Stage renvoie une exception pour tout appelant qui n'appartient pas au même sandbox de sécurité que le propriétaire de l'objet Stage (le fichier SWF principal). Pour plus d'informations, voir « Sécurité de la scène » à la page 1109.

Pour désactiver le mode plein écran pour les fichiers exécutés dans des navigateurs, les administrateurs peuvent définir `FullScreenDisable = 1` dans le fichier mms.cfg. Pour plus d'informations, voir « Contrôles administrateur » à la page 1092.

Pour accéder au mode plein écran dans un navigateur, un fichier SWF doit se trouver au sein d'une page HTML.

Sécurité du mode interactif plein écran

Flash Player 11.3 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player 11.3 et les versions ultérieures prennent en charge le mode interactif plein écran, dans lequel le contenu s'exécutant dans Flash Player peut remplir la totalité de l'écran *et prendre en charge la saisie de texte*. Pour accéder au mode interactif plein écran, la propriété `displayState` de la scène est définie sur la constante `StageDisplayState.FULL_SCREEN_INTERACTIVE`. Pour plus d'informations, voir « Utilisation du mode Plein écran » à la page 173.

L'exécution de fichiers SWF dans un sandbox distant impose la prise en considération de certains points de sécurité.

L'activation du mode plein écran s'effectue dans les balises `<object>` et `<embed>` de la page HTML qui contient la référence au fichier SWF. Pour ce faire, ajoutez le paramètre `allowFullScreenInteractive` et attribuez-lui la valeur `"true"` (la valeur par défaut est `"false"`), comme le montre l'exemple suivant :

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,
18,0"
    width="600" height="400" id="test" align="middle">
<param name="allowFullScreenInteractive" value="true" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowFullScreen="true" bgcolor="#333333"
    width="600" height="400"
    name="test" align="middle" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Une page HTML peut également utiliser un script pour générer des balises d'imbrication de fichiers SWF. Vous devez modifier le script de manière qu'il insère les paramètres `allowFullScreenInteractive` appropriés. Les pages HTML générées par Flash Professional et Flash Builder utilisent la fonction `AC_FL_RunContent()` pour incorporer des références aux fichiers SWF, et vous devez ajouter les paramètres `allowFullScreenInteractive`, comme dans l'exemple suivant :

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

Le code ActionScript qui lance le mode interactif plein écran peut être uniquement appelé en réponse à un événement souris ou clavier. S'il est appelé dans d'autres situations, Flash Player renvoie une exception.

Un message de superposition s'affiche lorsque le contenu passe en mode interactif plein écran. Ce message indique le domaine de la page plein écran, donne des instructions pour quitter le mode plein écran et fournit un bouton **Autoriser**. La superposition persiste jusqu'à ce que l'utilisateur clique sur **Autoriser** et accepte d'activer le mode interactif plein écran.

Pour désactiver le mode interactif plein écran pour les fichiers exécutés dans des navigateurs, les administrateurs peuvent définir `FullScreenInteractiveDisable = 1` dans le fichier mms.cfg. Pour plus d'informations, voir « Contrôles administrateur » à la page 1092.

Pour accéder au mode interactif plein écran dans un navigateur, un fichier SWF doit se trouver au sein d'une page HTML.

Chargement de contenu

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Le contenu Flash Player et AIR peut charger un grand nombre de types de contenus, notamment :

- Fichiers SWF
- Images
- Son
- Vidéo
- Fichiers HTML (AIR uniquement)
- JavaScript (AIR uniquement)

Chargement d'images et de fichiers SWF à l'aide de la classe Loader

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe Loader permet de charger des fichiers SWF et des images (fichiers JPG, GIF ou PNG). Un fichier SWF, s'il ne se trouve pas dans le sandbox local avec système de fichiers, peut charger des fichiers SWF et des images depuis n'importe quel domaine réseau. Seuls les fichiers SWF associés aux sandbox locaux peuvent charger des fichiers SWF et des images issus du système de fichiers local. Cependant, les fichiers du sandbox local avec réseau peuvent uniquement charger des fichiers SWF locaux qui se trouvent dans le sandbox local approuvé ou avec réseau. Les fichiers SWF associés au sandbox local avec réseau peuvent charger du contenu autre que des fichiers SWF (par exemple des images), mais ne peuvent pas accéder aux données du contenu chargé.

Lorsque vous chargez un fichier SWF d'une source non approuvée (telle qu'un domaine autre que celui du fichier SWF racine de l'objet Loader), il peut s'avérer utile de définir un masque pour ce dernier, afin d'empêcher le contenu chargé, qui est un enfant de l'objet Loader, d'apparaître dans des parties de la scène qui ne relèvent pas de ce masque, comme illustré par le code suivant :

```
import flash.display.*;
import flash.net.URLRequest;
var rect:Shape = new Shape();
rect.graphics.beginFill(0xFFFFFF);
rect.graphics.drawRect(0, 0, 100, 100);
addChild(rect);
var ldr:Loader = new Loader();
ldr.mask = rect;
var url:String = "http://www.unknown.example.com/content.swf";
var urlReq:URLRequest = new URLRequest(url);
ldr.load(urlReq);
addChild(ldr);
```

Lorsque vous appelez la méthode `load()` de l'objet Loader, vous pouvez spécifier un paramètre `context`, qui constitue un objet LoaderContext. La classe LoaderContext comporte trois propriétés qui permettent de définir le contexte d'utilisation du contenu chargé :

- `checkPolicyFile` : utilisez cette propriété uniquement pour le chargement d'un fichier image (pas pour un fichier SWF). Spécifiez-la pour un fichier image issu d'un domaine autre que celui du fichier contenant l'objet Loader. Si vous définissez cette propriété sur `true`, Loader recherche sur le serveur d'origine un fichier de régulation d'URL (voir « Contrôles de site Web (fichiers de régulation) » à la page 1095). Si le serveur autorise l'accès au domaine Loader, le code ActionScript des fichiers SWF du domaine Loader peut accéder à l'image chargée. En d'autres termes, vous pouvez utiliser soit la propriété `Loader.content` pour obtenir une référence à un objet Bitmap qui représente l'image chargée, soit la méthode `BitmapData.draw()` ou la méthode `BitmapData.drawWithQuality()` pour accéder aux pixels de l'image chargée. La méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures.
- `securityDomain` : utilisez cette propriété uniquement pour le chargement d'un fichier SWF (pas pour une image). Cette propriété peut être appelée pour un fichier SWF provenant d'un autre domaine que celui du fichier qui contient l'objet Loader. Seules les deux valeurs suivantes sont actuellement prises en charge par la propriété `securityDomain` : `null` (par défaut) et `SecurityDomain.currentDomain`. Si vous spécifiez `SecurityDomain.currentDomain`, le fichier SWF chargé est *importé* sur demande dans le sandbox du fichier SWF à l'origine du chargement. Par conséquent le fichier fonctionne comme s'il avait été chargé à partir du serveur du fichier appelant. Cette opération n'est permise que si le fichier de régulation d'URL se trouve sur le serveur du fichier SWF chargé, pour qu'il soit accessible au domaine du fichier SWF à l'origine du chargement. Si le fichier nécessaire est détecté, les deux fichiers peuvent librement effectuer une programmation croisée dès le début du chargement, puisqu'ils se trouvent dans le même sandbox. Notez que l'importation dans le sandbox peut presque être remplacée par un chargement ordinaire suivi d'un appel du fichier SWF chargé à la méthode `Security.allowDomain()`. Cette dernière peut s'avérer plus simple à utiliser puisque le fichier SWF chargé se trouve alors dans son sandbox naturel, pouvant ainsi accéder aux ressources de son propre serveur.
- `applicationDomain` : utilisez cette propriété uniquement lors du chargement d'un fichier SWF écrit dans ActionScript 3.0 (et non une image ou un fichier SWF écrit dans ActionScript 1.0 ou 2.0). Lors du chargement du fichier, vous pouvez spécifier s'il doit être placé dans un domaine d'application particulier, plutôt que dans le domaine par défaut, c'est-à-dire un nouveau domaine créé comme enfant du domaine d'application du fichier SWF à l'origine du chargement. Notez que les domaines d'application sont des sous-ensembles des domaines de sécurité. Ainsi, vous pouvez uniquement spécifier un domaine d'application cible si le fichier SWF chargé provient de votre propre de domaine de sécurité, soit parce qu'il appartient à votre propre serveur, soit parce que vous l'avez importé dans votre domaine de sécurité à l'aide de la propriété `securityDomain`. Si vous spécifiez un domaine d'application mais que le fichier SWF chargé fait partie d'un domaine de sécurité différent, le domaine que vous spécifiez dans `applicationDomain` est ignoré. Pour plus d'informations, voir « Utilisation de domaines d'application » à la page 152.

Pour plus d'informations, voir « Définition du contexte de chargement » à la page 207.

L'objet Loader possède une importante propriété, `contentLoaderInfo`, qui constitue un objet LoaderInfo. Contrairement à la plupart des objets, un objet LoaderInfo est partagé entre le fichier SWF à l'origine du chargement et le contenu chargé. Il est en outre accessible par les deux parties. Si le contenu chargé est un fichier SWF, il peut accéder à l'objet LoaderInfo au moyen de la propriété `DisplayObject.loaderInfo`. Les objets LoaderInfo incluent des informations telles que la progression du chargement, l'URL du fichier de chargement et du fichier chargé, la relation de confiance entre ces deux fichiers, et d'autres renseignements. Pour plus d'informations, voir « Surveillance de la progression du chargement » à la page 206.

Chargement de sons et vidéos

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

En dehors du contenu du sandbox local avec système de fichiers, tout contenu est autorisé à charger des éléments audio et vidéo en provenance d'un réseau grâce aux méthodes `Sound.load()`, `NetConnection.connect()` et `NetStream.play()`.

Seul le contenu du sandbox local avec système de fichiers et du sandbox d'application AIR peut charger des fichiers multimédias issus du système de fichiers local. Seul le contenu du sandbox local avec système de fichiers, du sandbox d'application AIR ou du sandbox approuvé localement peut accéder aux données de ces fichiers chargés.

D'autres restrictions s'appliquent à l'accès aux données à partir d'un média chargé. Pour plus d'informations, voir « Accès aux médias chargés comme s'il s'agissait de données » à la page 1111.

Chargement de fichiers SWF et d'images à l'aide de la balise <img> d'un champ de texte

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La balise `<img>` permet de charger des fichiers SWF et bitmap dans un champ de texte, comme le montre le code suivant :

```
<img src = 'filename.jpg' id = 'instanceName' >
```

Pour accéder au contenu chargé de cette manière, utilisez la méthode `getImageReference()` de l'occurrence de TextField, comme dans le code suivant :

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

Notez cependant que les fichiers SWF et images chargés de cette manière sont placés dans le sandbox correspondant à leur origine.

Lorsque vous chargez un fichier image à l'aide de la balise `<img>` d'un champ de texte, l'accès aux données de l'image peut être autorisé par le biais d'un fichier de régulation d'URL. Vous pouvez vérifier l'existence d'un tel fichier en ajoutant l'attribut `checkPolicyFile` à la balise `<img>`, comme le montre le code suivant :

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

Lorsque vous chargez un SWF à l'aide de la balise `<img>` d'un champ de texte, vous pouvez autoriser l'accès aux données de ce fichier SWF via un appel à la méthode `Security.allowDomain()`.

Si vous utilisez la balise `<img>` d'un champ de texte pour charger un fichier externe (plutôt que d'incorporer une classe Bitmap dans votre fichier SWF), un objet Loader est automatiquement créé comme enfant de l'objet TextField et le fichier externe est chargé dans l'objet Loader comme si vous aviez utilisé un tel objet en ActionScript pour charger ce fichier. Dans ce cas, la méthode `getImageReference()` renvoie l'objet Loader automatiquement créé. Aucune vérification de sécurité n'est nécessaire pour charger cet objet Loader car il se trouve dans le même sandbox de sécurité que le code appelant.

Toutefois, si vous faites référence à la propriété `content` de l'objet Loader pour accéder au média chargé, des règles de sécurité s'appliquent. Si le contenu est une image, vous devez implémenter un fichier de régulation d'URL ; s'il s'agit d'un fichier SWF, vous devez modifier le code de ce fichier de manière qu'il appelle la méthode `allowDomain()`.

Adobe AIR
Dans le sandbox de l'application, les balises <img> d'un champ de texte sont ignorées pour empêcher les attaques d'hameçonnage. Par ailleurs, le code exécuté dans le sandbox de l'application n'est pas autorisé à appeler la méthode de sécurité `allowDomain()`.

Contenu diffusé à l'aide de serveurs RTMP

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Media Server utilise le protocole RTMP (Real-Time Media Protocol) pour servir des données, des sons et des vidéos. Vous pouvez charger ces données multimédias par le biais de la méthode `connect()` de la classe NetConnection, en transmettant une URL RTMP en tant que paramètre. Flash Media Server peut restreindre les connexions et empêcher le téléchargement du contenu, selon le domaine du fichier requis. Pour plus d'informations, voir la documentation de Flash Media Server disponible en ligne à l'adresse suivante : www.adobe.com/go/learn_fms_docs_fr.

Pour extraire des graphiques d'exécution et des données audio de flux RTMP par le biais des méthodes `BitmapData.draw()`, `BitmapData.drawWithQuality()` et `SoundMixer.computeSpectrum()` , vous devez autoriser l'accès au serveur. Les propriétés ActionScript côté serveur `Client.videoSampleAccess` et `Client.audioSampleAccess` permettent d'accéder à des répertoires spécifiques de Flash Media Server. Pour plus d'informations, voir le Guide de référence ActionScript de Flash Media Server côté serveur. (La méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures.)

Programmation croisée

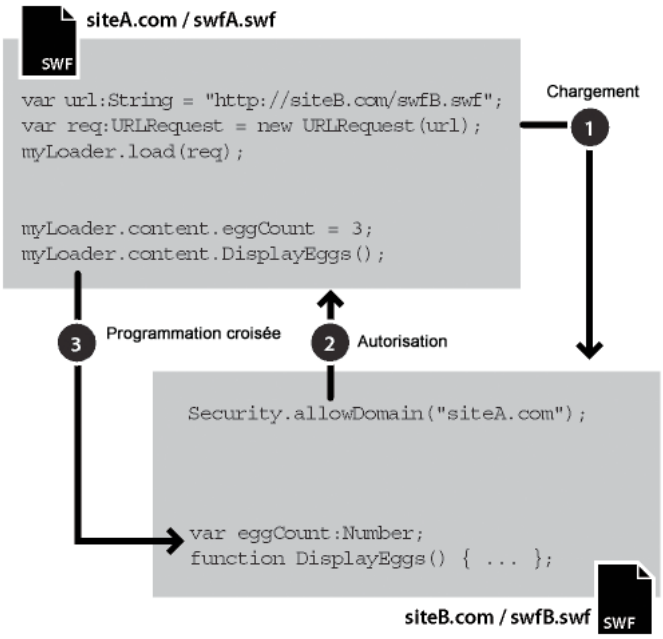
Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Si deux fichiers SWF écrits en ActionScript 3.0 ou deux fichiers HTML exécutés dans AIR sont transmis à partir d'un même domaine (supposons que l'URL du premier fichier SWF est http://www.example.com/swfA.swf et celle du second est http://www.example.com/swfB.swf), le code défini dans l'un des fichiers peut examiner et modifier les variables, objets, propriétés, méthodes, etc. de l'autre fichier, et inversement. On parle d'*intercodage*.

Si les deux fichiers sont issus de domaines différents (par exemple, http://siteA.com/swfA.swf et http://siteB.com/siteB.swf), Flash Player et AIR n'autorisent par défaut ni swfA.swf à créer un script pour swfB.swf, ni swfB.swf à créer un script pour swfA.swf. Un fichier SWF autorise les fichiers SWF provenant d'autres domaines en appelant `Security.allowDomain()`. Ainsi, en appelant `Security.allowDomain("siteA.com")`, swfB.swf accepte la programmation en provenance des fichiers SWF de siteA.com.

La programmation croisée n'est pas prise en charge entre les fichiers SWF AVM1 et AVM2. Un fichier SWF AVM1 est un fichier créé avec ActionScript 1.0 ou ActionScript 2.0 (AVM1 et AVM2 font référence à la machine virtuelle ActionScript). Vous pouvez néanmoins utiliser la classe LocalConnection pour échanger des données entre AVM1 et AVM2.

Dans tout contexte inter-domaines, il est important d'identifier clairement les parties impliquées. Dans le cadre de cette étude, le fichier effectuant la programmation croisée sera appelé *partie procédant à l'accès* (habituellement le fichier SWF procédant à l'accès), et l'autre côté sera appelé *partie cible* (généralement le fichier SWF cible). Lorsque siteA.swf programme siteB.swf, siteA.swf est la partie procédant à l'accès et site.B.swf la partie cible, comme le montre l'illustration suivante :



Les autorisations inter-domaines établies avec `Security.allowDomain()` sont asymétriques. Dans l'exemple précédent, siteA.swf peut programmer siteB.swf mais l'inverse n'est pas possible car siteA.swf n'a pas appelé la méthode `Security.allowDomain()` pour autoriser les fichiers SWF de siteB.com à le programmer. Vous pouvez définir des autorisations symétriques si les deux fichiers SWF appellent la méthode `Security.allowDomain()`.

En dehors de la protection des fichiers SWF contre les scripts inter-domaines provenant d'autres fichiers SWF, Flash Player protège également les fichiers SWF contre ce type de script provenant des fichiers HTML. La programmation HTML vers SWF est possible au moyen de rappels effectués avec la méthode `ExternalInterface.addCallback()`. Lorsque la programmation HTML vers SWF franchit les limites du domaine, le SWF cible doit également appeler `Security.allowDomain()`, comme s'il avait été appelé par un fichier SWF, faute de quoi l'opération échoue. Pour plus d'informations, voir « Contrôles de création (développeur) » à la page 1099.

Flash Player fournit en outre des contrôles de sécurité spécifiques à la programmation SWF vers HTML. Pour plus d'informations, voir « Contrôle de l'accès URL externe » à la page 1118.

Sécurité de la scène

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Certaines propriétés et méthodes de l'objet Stage sont disponibles pour tout sprite ou clip de la liste d'affichage.

Le premier fichier SWF chargé est cependant considéré comme le propriétaire de l'objet Stage. Par défaut, les propriétés et méthodes suivantes de l'objet Stage sont uniquement disponibles pour les fichiers SWF du même sandbox de sécurité que le propriétaire de l'objet Stage :

Propriétés	Méthodes
`align`	`addChild()`
displayState	`addChildAt()`
frameRate	`addEventListener()`
`height`	`dispatchEvent()`
`mouseChildren`	`hasEventListener()`
`numChildren`	`setChildIndex()`
`quality`	`willTrigger()`
`scaleMode`	
`showDefaultContextMenu`	
`stageFocusRect`	
`stageHeight`	
`stageWidth`	
`tabChildren`	
`textSnapshot`	
`width`	

Pour qu'un fichier SWF d'un sandbox différent de celui du propriétaire de l'objet Stage puisse accéder à ces propriétés et méthodes, le fichier SWF propriétaire de l'objet Stage doit appeler la méthode `Security.allowDomain()`. Pour plus d'informations, voir « Contrôles de création (développeur) » à la page 1099.

La propriété `frameRate` est un cas à part : tout fichier SWF peut lire la propriété `frameRate`. Toutefois, seuls les fichiers situés dans le sandbox de sécurité du propriétaire de l'objet Stage (ou ceux qui ont été autorisés à l'aide de la méthode `Security.allowDomain()`) peuvent modifier cette propriété.

Il existe également des restrictions sur les méthodes `removeChildAt()` et `swapChildrenAt()`, mais ce sont des restrictions différentes des autres. Pour appeler ces méthodes, le code ne doit pas se trouver dans le même domaine que le propriétaire de l'objet Stage mais dans le même domaine que le propriétaire du ou des objets enfant concernés ; en outre, les objets enfant peuvent appeler la méthode `Security.allowDomain()`.

Parcours de la liste d'affichage

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La capacité d'un fichier SWF d'accéder aux objets d'affichage chargés à partir d'autres sandbox fait l'objet de restrictions. Pour qu'un fichier SWF puisse accéder à un objet d'affichage créé par un autre fichier SWF dans un sandbox différent, le fichier SWF cible doit appeler la méthode `Security.allowDomain()` pour autoriser l'accès du domaine du fichier SWF procédant à l'appel. Pour plus d'informations, voir « Contrôles de création (développeur) » à la page 1099.

Pour accéder à un objet Bitmap chargé par un objet Loader, il faut qu'un fichier de régulation d'URL existe sur le serveur d'origine du fichier image et que ce fichier accorde une autorisation au domaine du fichier SWF qui essaie d'accéder à l'objet Bitmap (voir« Contrôles de site Web (fichiers de régulation) » à la page 1095).

L'objet LoaderInfo qui correspond au fichier chargé (et à l'objet Loader) inclut les trois propriétés suivantes, qui définissent la relation entre l'objet chargé et l'objet Loader : `childAllowsParent`, `parentAllowsChild` et `sameDomain`.

Sécurité des événements

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les événements liés à la liste d'affichage sont soumis à des restrictions d'accès de sécurité en fonction du sandbox de l'objet d'affichage qui distribue l'événement. Un événement de la liste d'affichage traverse des phases de capture et de propagation (voir « Gestion des événements » à la page 129). Au cours de ces deux phases un événement passe de l'objet d'affichage source aux objets d'affichage parent dans la liste d'affichage. Si un objet parent appartient à un sandbox de sécurité différent de celui de l'objet d'affichage source, la phase de capture ou de propagation vers le haut s'arrête en dessous de cet objet parent, sauf si une relation de confiance est établie entre le propriétaire de l'objet parent et celui de l'objet source. Cette confiance mutuelle s'établit des manières suivantes :

1. Le fichier SWF propriétaire de l'objet parent doit appeler la méthode `Security.allowDomain()` pour approuver le domaine du fichier SWF propriétaire de l'objet source.
2. Le fichier SWF propriétaire de l'objet source doit appeler la méthode `Security.allowDomain()` pour approuver le domaine du fichier SWF propriétaire de l'objet parent.

L'objet LoaderInfo qui correspond au fichier chargé (et à l'objet Loader) inclut les deux propriétés suivantes, qui définissent la relation entre l'objet chargé et l'objet Loader : `childAllowsParent` et `parentAllowsChild`.

Pour les événements distribués à partir d'objets autres que les objets d'affichage, il n'existe aucune vérification de sécurité ni aucune implication liée à la sécurité.

Accès aux médias chargés comme s'il s'agissait de données

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour accéder aux données chargées, faites appel aux méthodes `BitmapData.draw()`, `BitmapData.drawWithQuality()` et `SoundMixer.computeSpectrum()`. Par défaut, il est impossible d'obtenir des données de pixels ou des données audio d'objets graphiques ou audio rendus ou lus par un fichier multimédia chargé dans un autre sandbox. Vous disposez toutefois des méthodes suivantes pour accorder un droit d'accès à ces données en franchissant les limites du sandbox :

- Dans le contenu en cours de rendu ou lors de la lecture de données auxquelles vous voulez accéder, appelez la méthode `Security.allowDomain()` pour accorder un droit d'accès au contenu hébergé dans d'autres domaines.
- Pour un fichier image, son ou vidéo chargé, ajoutez un fichier de régulation d'URL sur le serveur du fichier chargé. Ce fichier de régulation doit accorder l'accès au domaine du fichier SWF qui tente d'appeler la méthode `BitmapData.draw()`, `BitmapData.drawWithQuality()` ou `SoundMixer.computeSpectrum()` pour extraire des données de ce fichier. La méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures.

Les sections qui suivent offrent des détails sur l'accès aux données bitmap, son et vidéo.

Accès aux données bitmap

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les méthodes `draw()` et `drawWithQuality()` (Flash Player 11.3 et AIR 3.3) d'un objet BitmapData permettent d'extraire les pixels actuellement affichés de tout objet d'affichage vers l'objet BitmapData. Il peut s'agir des pixels d'un objet MovieClip, d'un objet Bitmap ou d'un objet d'affichage. Vous devez remplir les conditions suivantes pour que ces méthodes dessinent des pixels sur l'objet BitmapData :

- Si l'objet source n'est pas un fichier bitmap chargé, l'objet source et (dans le cas d'un objet Sprite ou MovieClip) tous ses objets enfant doivent provenir du même domaine que l'objet appelant la méthode draw ou se trouver dans un fichier SWF qui est devenu accessible à l'objet appelant suite à l'appel de la méthode `Security.allowDomain()`.
- Si l'objet source est un fichier bitmap chargé, cet objet doit provenir du même domaine que l'objet appelant la méthode draw ou son serveur source doit inclure un fichier de régulation d'URL qui accorde l'autorisation nécessaire au domaine appelant.

Si ces conditions ne sont pas réunies, une exception SecurityError est renvoyée.

Lorsque vous appelez la méthode `load()` de la classe Loader, vous pouvez spécifier un paramètre `context`, qui constitue un objet LoaderContext. Si vous réglez la propriété `checkPolicyFile` de l'objet LoaderContext sur `true`, Flash Player recherche un fichier de régulation d'URL sur le serveur à partir duquel l'image est chargée. S'il existe un fichier de régulation autorisant le domaine du fichier SWF à l'origine du chargement, le fichier peut accéder aux données de l'objet Bitmap ; dans le cas contraire, l'accès est refusé.

Vous pouvez également spécifier une propriété `checkPolicyFile` dans une image chargée via la balise `<img>` d'un champ de texte. Pour plus d'informations, voir « Chargement de fichiers SWF et d'images à l'aide de la balise <img> d'un champ de texte » à la page 1106.

Accès aux données audio

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les API ActionScript 3.0 suivantes, liées aux sons, font l'objet de restrictions de sécurité :

- Méthode `SoundMixer.computeSpectrum()` : toujours autorisée pour le code qui s'exécute dans le même sandbox de sécurité que le fichier audio. Des contrôles de sécurité sont nécessaires pour le code qui s'exécute dans d'autres sandbox.
- Méthode `SoundMixer.stopAll()` : toujours autorisée pour le code qui s'exécute dans le même sandbox de sécurité que le fichier audio. Des contrôles de sécurité sont nécessaires pour les fichiers se trouvant dans d'autres sandbox.
- Propriété `id3` de l'objet Sound : toujours autorisée pour les fichiers SWF qui se trouvent dans le même sandbox de sécurité que le fichier son. Des contrôles de sécurité sont nécessaires pour le code qui s'exécute dans d'autres sandbox.

Chaque son est associé à deux types de sandbox, un sandbox de contexte et un sandbox de propriétaire :

- Le domaine d'origine du son détermine le sandbox de contexte. Celui-ci établit si les données peuvent être extraites du son via la propriété `id3` du son et la méthode `SoundMixer.computeSpectrum()`.
- L'objet qui déclenche la lecture du son détermine le sandbox de propriétaire, qui établit à son tour si le son peut être arrêté à l'aide de la méthode `SoundMixer.stopAll()`.

Lorsque vous chargez le son à l'aide de la méthode `load()` de la classe Sound, vous pouvez spécifier un paramètre `context`, qui constitue un objet SoundLoaderContext. Si vous définissez la propriété `checkPolicyFile` de l'objet SoundLoaderContext sur `true`, le moteur d'exécution recherche un fichier de régulation d'URL sur le serveur à partir duquel est chargé l'audio. S'il existe un fichier de régulation autorisant le domaine du code à l'origine du chargement, le code peut accéder à la propriété `id` de l'objet Sound ; dans le cas contraire, l'accès est refusé. En outre, la propriété `checkPolicyFile` peut permettre d'activer la méthode `SoundMixer.computeSpectrum()` pour les sons chargés.

La méthode `SoundMixer.areSoundsInaccessible()` vous permet de savoir si l'appel à la méthode `SoundMixer.stopAll()` n'entraînerait pas l'arrêt de tous les sons parce que le sandbox de l'une ou de plusieurs des propriétés d'objet son est inaccessible à l'appelant.

La méthode `SoundMixer.stopAll()` permet d'arrêter tous les sons dont le sandbox de propriétaire est le même que celui de l'appelant de `stopAll()`. Elle arrête également les sons dont la lecture a été déclenchée par des fichiers SWF ayant appelé la méthode `Security.allowDomain()` pour autoriser le domaine du fichier SWF appelant la méthode `stopAll()`. Tous les autres sons ne sont pas arrêtés ; vous pouvez vérifier leur présence en appelant la méthode `SoundMixer.areSoundsInaccessible()`.

L'appel de la méthode `computeSpectrum()` demande que chaque son en cours de lecture soit issu du même sandbox que l'objet appelant la méthode ou de la même source qui a autorisé l'accès au sandbox de l'appelant. Autrement, une exception SecurityError est renvoyée. Pour les sons chargés à partir de sons incorporés dans la bibliothèque d'un fichier SWF, l'autorisation est accordée en appelant la méthode `Security.allowDomain()` dans le fichier SWF chargé. Pour les données audio chargées à partir de sources autres que des fichiers SWF (issues de fichiers mp3 chargés ou de fichiers vidéo), un fichier de régulation d'URL hébergé sur le serveur source doit autoriser l'accès aux données figurant dans le fichier multimédia chargé.

Pour plus d'informations, voir « Contrôles de création (développeur) » à la page 1099 et « Contrôles de site Web (fichiers de régulation) » à la page 1095.

Pour accéder aux données audio à partir de flux RTMP, vous devez autoriser l'accès au serveur. La propriété ActionScript côté serveur `Client.audioSampleAccess` permet d'accéder à des répertoires spécifiques de Flash Media Server. Pour plus d'informations, voir le Guide de référence ActionScript de Flash Media Server côté serveur.

Accès aux données vidéo

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `BitmapData.draw()` ou `BitmapData.drawWithQuality()` permet de capturer des données de pixels à partir de l'image active d'une vidéo. (La méthode `drawWithQuality` est disponible dans Flash Player 11.3 et les versions ultérieures et dans AIR 3.3 et les versions ultérieures.)

Il existe deux types de vidéo :

- La vidéo diffusée en continu via RTMP à partir de Flash Media Server
- La vidéo progressive, chargée à partir d'un fichier FLV ou F4V

Pour extraire les graphiques d'exécution de flux RTMP par le biais des méthodes draw, vous devez autoriser l'accès au serveur. La propriété ActionScript côté serveur `Client.videoSampleAccess` permet d'accéder à des répertoires spécifiques de Flash Media Server. Pour plus d'informations, voir le Guide de référence ActionScript de Flash Media Server côté serveur.

Lorsque vous appelez la méthode draw avec la vidéo progressive comme paramètre `source`, l'appelant de la méthode doit provenir du même sandbox que le fichier FLV ou le serveur du fichier FLV doit contenir un fichier de régulation qui autorise le domaine du fichier SWF appelant. Pour demander le téléchargement du fichier de régulation, définissez la propriété `checkPolicyFile` de l'objet NetStream sur `true`.

Chargement des données

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Un contenu Flash Player et AIR peut échanger des données avec les serveurs. Le chargement de données et le chargement de fichier multimédia sont deux opérations distinctes, car les informations chargées apparaissent sous forme d'objets de programme, au lieu d'être affichées sous forme de fichier multimédia. En règle générale, un contenu peut charger des données issues du même domaine que le domaine d'origine du contenu. Un contenu requiert cependant le plus souvent des fichiers de régulation pour charger les données issues d'autres domaines (voir « Contrôles de site Web (fichiers de régulation) » à la page 1095).

Remarque : *étant donné qu'un contenu qui s'exécute dans le sandbox d'application AIR n'est jamais transmis à partir d'un domaine distant (sauf si le développeur importe intentionnellement un contenu distant dans le sandbox d'application), il ne peut pas participer aux types d'attaques bloqués par les fichiers de régulation. Un contenu AIR qui réside dans le sandbox d'application est autorisé à charger des données par les fichiers de régulation. Toutefois, un contenu AIR qui réside dans d'autres sandbox est soumis aux restrictions décrites dans cette section.*

Utilisation de URLLoader et URLStream

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Vous pouvez charger des données telles que des fichiers XML ou texte. Les méthodes `load()` des classes URLLoader et URLStream sont régies par les autorisations d'un fichier de régulation d'URL.

Si vous utilisez la méthode `load()` pour charger un contenu issu d'un autre domaine que celui du code qui appelle la méthode, le moteur d'exécution recherche le fichier de régulation d'URL sur le serveur des actifs chargés. S'il existe un fichier de régulation et qu'il autorise l'accès au domaine du contenu à l'origine du chargement, vous pouvez charger les données.

Connexion aux sockets

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Par défaut, le moteur d'exécution recherche un fichier de régulation socket transmis à partir du port 843. Comme dans le cas des fichiers de régulation d'URL, ce fichier est appelé *Fichier de régulation maître*.

Lors de l'introduction des fichiers de régulation dans Flash Player 6, les fichiers socket n'étaient pas pris en charge. Les connexions aux serveurs socket étaient autorisées par un fichier de régulation figurant à l'emplacement par défaut sur un serveur HTTP à condition d'utiliser le port 80 du même hôte que le serveur socket. Flash Player 9 prend toujours cette fonctionnalité en charge mais ce n'est pas le cas de Flash Player 10. Dans Flash Player 10, seuls les fichiers de régulation socket peuvent autoriser les connexions socket.

A l'instar des fichiers de régulation d'URL, les fichiers de régulation socket prennent en charge une instruction de méta-régulation qui identifie les ports dédiés aux fichiers de régulation. La méta-régulation des fichiers de régulation socket est définie sur « all », plutôt que sur « master-only ». Par conséquent, à moins qu'un paramètre plus restrictif soit défini dans le fichier de régulation maître, Flash Player considère comme acquis que n'importe quel socket de l'hôte peut servir un fichier de régulation socket.

L'accès aux connexions socket et socket XML est désactivé par défaut, même si le socket auquel vous vous connectez se trouve dans le même domaine que le fichier SWF. Vous pouvez autoriser l'accès de niveau socket en servant un fichier de régulation socket à partir d'un des emplacements suivants :

- Le port 843 (emplacement du fichier de régulation maître)
- Le même port que la connexion socket principale
- Un autre port que la connexion socket principale

Par défaut, Flash Player recherche un fichier de régulation socket sur le port 843 et sur le même port que la connexion socket principale. Si vous souhaitez servir un fichier de régulation socket à partir d'un autre port, le fichier SWF doit appeler `Security.loadPolicyFile()`.

La syntaxe d'un fichier de régulation socket est identique à celle d'un fichier de régulation d'URL, à la différence qu'elle doit aussi spécifier les ports accessibles. Un fichier de régulation socket servi via un port dont le numéro est inférieur à 1024 peut autoriser l'accès à tous les ports. Un fichier de régulation transmis via le port 1024 ou supérieur ne peut définir l'accès qu'au port 1024 et aux ports supérieurs. Les ports accessibles sont spécifiés par l'attribut `to-ports` dans la balise `<allow-access-from>`. Il est possible d'utiliser des numéros de ports, des plages de ports et des caractères génériques.

Voici un exemple de fichier de régulation socket :

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<!-- Policy file for xmlsocket://socks.mysite.com -->
<cross-domain-policy>
    <allow-access-from domain="*" to-ports="507" />
    <allow-access-from domain="*.example.com" to-ports="507,516" />
    <allow-access-from domain="*.example.org" to-ports="516-523" />
    <allow-access-from domain="adobe.com" to-ports="507,516-523" />
    <allow-access-from domain="192.0.34.166" to-ports="*" />
</cross-domain-policy>
```

Pour récupérer un fichier de régulation socket sur le port 843 ou sur le même port que la connexion socket principale, appelez la méthode `Socket.connect()` ou `XMLSocket.connect()`. Flash Player recherche d'abord un fichier de régulation maître sur le port 843. S'il en trouve un, il vérifie s'il contient une instruction de méta-régulation interdisant les fichiers de régulation socket sur le port cible. Si l'accès n'est pas interdit, Flash Player recherche l'instruction `allow-access-from` appropriée dans le fichier de régulation maître. En l'absence d'une telle instruction, il recherche un fichier de régulation sur le même port que la connexion socket principale.

Pour récupérer un fichier de régulation socket à un autre emplacement, appelez d'abord la méthode `Security.loadPolicyFile()` en utilisant la syntaxe `"xmlsocket"` spéciale, comme illustré ci-dessous :

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

Appelez la méthode `Security.loadPolicyFile()` avant d'appeler `Socket.connect()` ou `XMLSocket.connect()`. Flash Player peut alors attendre le retour de votre requête de fichier de régulation avant de décider ou non d'autoriser la connexion principale. Cependant, si le fichier de régulation maître spécifie que l'emplacement cible ne peut pas servir des fichiers de régulation, l'appel à `loadPolicyFile()` est sans effet, même si un fichier de régulation se trouve à l'emplacement en question.

Si vous implémentez un serveur socket et que vous devez fournir un fichier de régulation socket, vous devez choisir entre fournir le fichier de régulation sur le port qui accepte les connexions principales et utiliser un autre port. Dans les deux cas, votre serveur doit attendre la première transmission en provenance de votre client avant d'envoyer une réponse.

Lorsque Flash Player demande un fichier de régulation, il transmet toujours la chaîne suivante dès que la connexion est établie :

```
<policy-file-request/>
```

Une fois que le serveur a reçu la chaîne, il peut transmettre le fichier de régulation. La requête émise par Flash Player se termine toujours par un octet nul et la réponse du serveur doit se terminer de même.

N'envisagez pas d'utiliser la même connexion pour la requête de fichier de régulation et pour la connexion principale. Vous devez fermer la connexion une fois le fichier de régulation transmis. A défaut, Flash Player ferme la connexion du fichier de régulation, puis établit une autre connexion pour la connexion principale.

Protection des données

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour protéger les données contre toute écoute ou modification lorsqu'elles transitent via Internet, vous disposez des protocoles TLS (Transport Layer Security) et SSL (Socket Layer Security) sur le serveur d'origine des données. Vous pouvez ensuite établir une connexion au serveur via le protocole HTTPS.

Dans les applications destinées à AIR 2 ou ultérieur, vous pouvez également protéger les communications socket TCP. La classe SecureSocket permet de lancer une connexion socket à un serveur socket qui utilise TLS version 1 or SSL version 4.

Envoi de données

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Il se produit un envoi de données lorsque le code envoie des données à un serveur ou une ressource. L'envoi de données est systématiquement autorisé si le contenu est issu d'un domaine du réseau. Un fichier SWF local peut envoyer des données à des adresses réseau uniquement si elles se trouvent dans le sandbox approuvé localement, le sandbox local avec accès au réseau ou le sandbox d'application AIR. Pour plus d'informations, voir « Sandbox locaux » à la page 1087.

Vous pouvez utiliser la fonction `flash.net.sendToURL()` pour envoyer des données à une URL. D'autres méthodes permettent d'envoyer des requêtes aux URL. Il s'agit notamment des méthodes de chargement, telles que `Loader.load()` et `Sound.load()`, et des méthodes de chargement de données, telles que `URLLoader.load()` et `URLStream.load()`.

Chargement et téléchargement de fichiers

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La méthode `FileReference.upload()` lance le chargement d'un fichier sélectionné par l'utilisateur vers un serveur distant. Vous devez appeler la méthode `FileReference.browse()` ou `FileReferenceList.browse()` avant la méthode `FileReference.upload()`.

Le code qui lance la méthode `FileReference.browse()` ou `FileReferenceList.browse()` ne peut être appelé qu'en réponse à un événement de souris ou de clavier. S'il est appelé dans d'autres situations, Flash Player 10 et les versions ultérieures renvoie une exception. Un événement lancé par l'utilisateur n'est toutefois pas nécessaire pour appeler ces méthodes à partir du sandbox d'application AIR.

L'appel de la méthode `FileReference.download()` ouvre une boîte de dialogue dans laquelle l'utilisateur peut télécharger un fichier à partir d'un serveur distant.

Remarque : *si votre serveur nécessite une authentification des utilisateurs, seuls les fichiers SWF s'exécutant dans un navigateur (c'est-à-dire utilisant le plug-in du navigateur ou un contrôle ActiveX) peuvent fournir une boîte de dialogue pour demander à l'utilisateur un nom et un mot de passe d'authentification, ceci uniquement pour les téléchargements. Flash Player ne permet pas de charger des fichiers sur un serveur qui nécessite une authentification utilisateur.*

Les chargements et téléchargements ne sont autorisés que si le fichier SWF appelant appartient au sandbox local avec système de fichiers.

Par défaut, un fichier SWF ne peut pas lancer un chargement ou un téléchargement avec un serveur autre que le sien. Il peut le faire si le serveur en question fournit un fichier de régulation accordant un accès au domaine du fichier SWF appelant.

Chargement de contenu incorporé à partir de fichiers SWF importés dans un domaine de sécurité

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsque vous chargez un fichier SWF, vous pouvez définir le paramètre `context` de la méthode `load()` de l'objet Loader utilisé pour le chargement. Ce paramètre prend un objet LoaderContext. Si vous réglez la propriété `securityDomain` de cet objet LoaderContext sur `Security.currentDomain`, Flash Player recherche un fichier de régulation d'URL sur le serveur à partir duquel le fichier SWF est chargé. S'il en existe un qui autorise l'accès au domaine du fichier SWF à l'origine du chargement, vous pouvez charger le fichier SWF sous forme de média importé. De cette manière, le fichier à l'origine du chargement obtient l'accès aux objets de la bibliothèque du fichier SWF.

Une autre méthode permet d'autoriser l'accès d'un fichier SWF aux classes des fichiers SWF chargés à partir d'un autre sandbox de sécurité : le fichier SWF chargé doit simplement appeler la méthode `Security.allowDomain()` pour autoriser l'accès du domaine du fichier SWF appelant. Cet appel à la méthode `Security.allowDomain()` peut s'ajouter à la méthode constructeur de la classe principale du fichier SWF chargé. Ensuite, le fichier SWF à l'origine du chargement doit ajouter un écouteur d'événement pour répondre à l'événement `init` distribué par la propriété `contentLoaderInfo` de l'objet Loader. Cet événement est distribué lorsque le fichier SWF chargé a appelé la méthode `Security.allowDomain()` dans la méthode constructeur et que des classes du fichier SWF chargé sont disponibles pour le fichier SWF à l'origine du chargement. Le fichier SWF en cours de chargement peut extraire les classes du fichier SWF chargé en appelant `Loader.contentLoaderInfo.applicationDomain.getDefinition()` ou Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames() (Flash Player 11.3 et les versions ultérieures et AIR 3.3 et les versions ultérieures).

Utilisation de contenus existants

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Dans Flash Player 6, le domaine utilisé par certains paramètres Flash Player dépend de la fin du domaine du fichier SWF. Il s'agit notamment des paramètres d'autorisations relatifs à la caméra et au microphone, aux quotas de stockage et au stockage d'objets partagés persistants.

Si le domaine du fichier SWF comprend plus de deux segments, par exemple www.example.com, le premier segment du domaine (www) est supprimé et la fin du domaine est exploitée. Ainsi, dans Flash Player 6, www.exemple.com et magasin.exemple.com ont en commun le domaine « example.com » pour ces paramètres. De même, www.exemple.co.fr et magasin.exemple.co.fr ont tous les deux recours au domaine exemple.co.fr pour ces paramètres. Cette caractéristique pose problème pour les fichiers SWF issus de domaines distincts, tels que example1.co.uk et example2.co.uk, qui ont alors accès aux mêmes objets partagés.

A compter de Flash Player 7, les paramètres du lecteur sont choisis par défaut en fonction du domaine exact d'un fichier SWF. Par exemple, le fichier SWF de www.exemple.com applique les paramètres du lecteur de www.exemple.com, et le fichier SWF de magasin.exemple.com utiliserait les paramètres différents de magasin.exemple.com.

Dans le cas d'un fichier SWF écrit en ActionScript 3.0, si `Security.exactSettings` conserve la valeur par défaut `true`, Flash Player utilise les domaines exacts pour les paramètres de lecteur. Si sa valeur est `false`, Flash Player utilise les paramètres de domaine de Flash Player 6. Si vous modifiez la valeur de `exactSettings`, vous devez le faire avant que ne survienne tout événement obligeant Flash Player à choisir des paramètres de lecteur (par exemple l'utilisation d'une caméra ou d'un microphone, ou l'extraction d'un objet partagé persistant).

Si vous avez publié un fichier SWF avec la version 6 et créé des objets partagés persistants à partir de ce fichier, vous devez, pour récupérer ces objets persistants à partir d'un fichier SWF en ActionScript 3.0, attribuer la valeur `false` à `Security.exactSettings` avant d'appeler `SharedObject.getLocal()`.

Définition des autorisations LocalConnection

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

La classe LocalConnection permet d'envoyer des messages entre deux applications Flash Player ou AIR. Les objets LocalConnection ne communiquent qu'entre contenu Flash Player ou AIR qui s'exécute sur le même ordinateur client. Ils peuvent toutefois s'exécuter dans des applications distinctes. Ainsi, un fichier SWF qui s'exécute dans un navigateur, un fichier SWF qui s'exécute en mode Projection et une application AIR peuvent tous communiquer par le biais de la classe LocalConnection.

A chaque communication LocalConnection correspond un émetteur et un récepteur. Par défaut, Flash Player permet les communications LocalConnection si le code s'exécute dans le même domaine. Si le code s'exécute dans des sandbox distincts, le récepteur doit accorder une autorisation à l'émetteur à l'aide de la méthode `LocalConnection.allowDomain()`. La chaîne passée comme argument à la méthode `LocalConnection.allowDomain()` peut contenir n'importe lesquels des éléments suivants : noms de domaine exacts, adresses IP et caractère générique `*`.

Le format de la méthode `allowDomain()` n'est plus le même que dans ActionScript 1.0 et 2.0. Dans ces versions, `allowDomain` était une méthode de rappel que vous implémentiez. Dans ActionScript 3.0, `allowDomain()` est une méthode intégrée de la classe LocalConnection que vous appelez. Le fonctionnement de la nouvelle version de `allowDomain()` est semblable à celui de `Security.allowDomain()`.

Un fichier SWF peut utiliser la propriété `domain` de la classe LocalConnection pour déterminer le domaine.

Contrôle de l'accès URL externe

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Les API suivantes gèrent la génération de scripts et l'accès URL externes (par le biais d'URL HTTP, de mailto:, etc.) :

- La fonction `flash.system.fscommand()`
- La méthode `ExternalInterface.call()`
- La fonction `flash.net.navigateToURL()`

Si un contenu est chargé à partir du système de fichiers local, les appels à ces méthodes n'aboutissent que si le code et la page Web qui le contient (le cas échéant) se trouvent dans le sandbox approuvé localement ou le sandbox d'application AIR. Ils échouent si le contenu provient du sandbox local avec accès au réseau ou local avec système de fichiers.

Si le contenu n'est pas chargé localement, toutes ces API peuvent communiquer avec la page Web à laquelle elles sont intégrées, selon la valeur du paramètre AllowScriptAccess décrit ci-dessous. La fonction `flash.net.navigateToURL()` peut en outre communiquer avec toute fenêtre ou tout cadre de navigateur ouvert, pas seulement avec la page contenant le fichier SWF. Pour plus d'informations à ce sujet, voir « Utilisation de la fonction navigateToURL() » à la page 1119.

Le paramètre `AllowScriptAccess` du code HTML qui charge un fichier SWF permet de contrôler l'accès URL externe à partir du fichier SWF. Définissez ce paramètre dans la balise PARAM ou EMBED. Si la valeur du paramètre `AllowScriptAccess` n'est pas définie, le fichier SWF et la page HTML peuvent uniquement communiquer s'ils appartiennent au même domaine.

Le paramètre `AllowScriptAccess` prend en charge trois valeurs : `"always"`, `"sameDomain"` et `"never"`.

- Lorsque le paramètre `AllowScriptAccess` est défini sur `"always"`, le fichier SWF peut communiquer avec la page HTML à laquelle il est intégré, même s'il ne provient pas du même domaine qu'elle.
- Lorsque le paramètre `AllowScriptAccess` est défini sur `"sameDomain"`, le fichier SWF peut communiquer avec la page HTML à laquelle il est intégré, uniquement s'il provient du même domaine. Il s'agit de la valeur par défaut du paramètre `AllowScriptAccess`. Utilisez ce réglage ou ne définissez pas la valeur du paramètre `AllowScriptAccess` pour empêcher un fichier SWF d'un domaine d'accéder à un script sur une page HTLM issue d'un autre domaine.
- Lorsque le paramètre `AllowScriptAccess` est défini sur `"never"`, le fichier SWF ne peut communiquer avec aucune page HTML. L'utilisation de cette valeur est désapprouvée depuis la sortie d'Adobe Flash CS4 Professional. Elle n'est pas recommandée et elle ne devrait pas s'avérer nécessaire si vous ne servez pas de fichiers SWF non approuvés à partir de votre propre domaine. Si vous devez envoyer des fichiers SWF non approuvés, Adobe vous conseille de créer un sous-domaine distinct et d'y placer l'ensemble du contenu non approuvé.

L'exemple suivant illustre le réglage de la balise `AllowScriptAccess` dans une page HTML pour autoriser l'accès URL externe à un autre domaine :

```
<object id='MyMovie.swf' classid='clsid:D27CDB6E-AE6D-11cf-96B8-444553540000'
codebase='http://download.adobe.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0'
height='100%' width='100%'>
<param name='AllowScriptAccess' value='always'/>
<param name='src' value=''MyMovie.swf'/>
<embed name='MyMovie.swf' pluginspage='http://www.adobe.com/go/getflashplayer'
src='MyMovie.swf' height='100%' width='100%' AllowScriptAccess='never'/>
</object>
```

Utilisation de la fonction navigateToURL()

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Outre le réglage de sécurité spécifié par le paramètre `allowScriptAccess` (voir plus haut), la fonction `navigateToURL()` propose un autre paramètre facultatif : `target`. Le paramètre `target` permet de spécifier le nom d'une fenêtre ou d'un cadre HTML auquel envoyer la requête URL. D'autres restrictions de sécurité s'appliquent à de telles requêtes et elles varient selon que la fonction `navigateToURL()` est utilisée comme instruction de programmation ou non.

Pour les instructions de programmation, telles que `navigateToURL("javascript: alert('Hello from Flash Player.')")`, les règles suivantes s'appliquent :

- Si le fichier SWF est un fichier local approuvé, la requête aboutit.
- Si la cible est la page HTML à laquelle le fichier SWF est intégré, les règles `allowScriptAccess` décrites plus haut s'appliquent.
- Si la cible contient du contenu issu du même domaine que le fichier SWF, la requête aboutit.
- Si la cible contient du contenu issu d'un autre domaine que le fichier SWF et qu'aucune des deux conditions précédentes n'est remplie, la requête échoue.

Pour les instructions autres que les instructions de programmation (telles que HTTP, HTTPS et `mailto:`), la requête échoue si toutes les conditions suivantes sont réunies :

- La cible correspond à un des mots clés spéciaux `"_top"` ou `"_parent"`
- Le fichier SWF se trouve dans une page Web hébergée dans un domaine différent et
- Le paramètre `allowScriptAccess` n'est pas défini sur `"always"` dans le fichier SWF

Pour plus d'informations

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Pour plus d'informations sur l'accès URL externe, voir les sections suivantes du manuel *Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash* :

- Fonction flash.system.fscommand()
- Méthode call() de la classe ExternalInterface
- Fonction `flash.net.navigateToURL()`

Objets partagés

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Flash Player offre la possibilité d'utiliser des *objets partagés*, c'est-à-dire des objets ActionScript qui persistent en dehors d'un fichier SWF, soit localement dans le système de fichiers de l'utilisateur, soit à distance sur un serveur RTMP. Les objets partagés, comme d'autres médias dans Flash Player, sont répartis dans des sandbox de sécurité. Le modèle de sandbox des objets partagés s'avère toutefois différent parce que les objets partagés ne sont pas des ressources accessibles depuis d'autres domaines. Les objets partagés sont au contraire extraits d'un magasin d'objets partagés propre au domaine de chaque fichier SWF qui appelle les méthodes de la classe SharedObject. En règle générale, un magasin d'objets partagés est encore plus spécifique que le domaine d'un fichier SWF : par défaut, chaque fichier SWF utilise un magasin d'objets partagés propre à la totalité de son URL d'origine. Pour plus d'informations sur les objets partagés, voir « Objets partagés » à la page 728.

Le fichier SWF peut utiliser le paramètre `localPath` des méthodes `SharedObject.getLocal()` et `SharedObject.getRemote()` afin d'exploiter un magasin d'objets partagés associé à une partie de son URL seulement. Ainsi, le fichier SWF peut autoriser le partage avec d'autres fichiers SWF issus d'autres URL. Même si la valeur transmise au paramètre `localPath` est `'/'` , celle-ci spécifie tout de même un magasin d'objets partagés propres à son domaine.

Les utilisateurs peuvent limiter l'accès aux objets partagés via la boîte de dialogue Paramètres de Flash Player ou via le Gestionnaire des paramètres. Par défaut, le volume d'objets partagés créés ne peut dépasser 100 Ko de données par domaine. Les administrateurs et les utilisateurs peuvent également limiter la capacité à écrire dans le système de fichiers. Pour plus d'informations, voir « Contrôles administrateur » à la page 1092 et « Contrôles utilisateur » à la page 1093.

Vous pouvez définir la sécurisation d'un objet partagé en attribuant la valeur `true` au paramètre `secure` de la méthode `SharedObject.getLocal()` ou `SharedObject.getRemote()`. Notez les points suivants concernant le paramètre `secure` :

- Si ce paramètre est défini sur `true`, Flash Player crée un nouvel objet partagé sécurisé ou obtient une référence à un objet partagé sécurisé existant. Cet objet sécurisé partagé peut uniquement être lu par des fichiers SWF ou écrit dans des fichiers SWF reçus via HTTPS appelant `SharedObject.getLocal()` avec le paramètre `secure` défini sur `true`.
- Si ce paramètre est défini sur `false`, Flash Player crée un objet partagé ou obtient une référence à un objet partagé existant, qui peut être lu ou écrit par des fichiers SWF reçus via des connexions autres que HTTPS.

Si le fichier SWF appelant ne provient pas d'une URL HTTPS, l'attribution de la valeur `true` au paramètre `secure` de la méthode `SharedObject.getLocal()` ou `SharedObject.getRemote()` renvoie une exception SecurityError.

Le choix du magasin d'objets partagé dépend de l'URL d'origine du fichier SWF. Cela reste vrai même dans les deux cas de figure où un fichier SWF ne provient pas d'une URL simple : importation et chargement dynamique. L'importation s'applique lorsque vous chargez un fichier SWF dont la propriété `LoaderContext.securityDomain` a la valeur `SecurityDomain.currentDomain`. Dans ce cas, le fichier SWF chargé portera une pseudo-URL qui commence par le domaine du fichier SWF à l'origine du chargement, puis spécifie sa véritable URL d'origine. Le chargement dynamique renvoie au chargement d'un fichier SWF à l'aide de la méthode `Loader.loadBytes()`. Dans cette situation, le fichier SWF chargé porte une pseudo-URL qui commence par l'URL complète du fichier SWF à l'origine du chargement, suivie d'un entier d'identification. Dans les deux cas de figure (importation et chargement dynamique), la pseudo-URL du fichier SWF peut être analysée à l'aide de la propriété `LoaderInfo.url`. Cette pseudo-URL est traitée comme une URL réelle pour définir le magasin d'objets partagés. Elle peut être utilisée en partie ou dans son intégralité comme paramètre `localPath` d'un objet partagé.

Les utilisateurs et les administrateurs peuvent choisir de désactiver l'utilisation d'*objets partagés tiers*. Il s'agit des objets partagés utilisés par tout fichier SWF exécuté dans un navigateur Web lorsque l'URL d'origine de ce fichier SWF est d'un domaine différent de l'URL affichée dans la barre d'adresse du navigateur. Les administrateurs et utilisateurs peuvent choisir de désactiver l'utilisation des objets partagés tiers pour des raisons de confidentialité, s'ils souhaitent éviter la surveillance inter-domaines. Pour éviter cette restriction, il est judicieux de veiller à ce que tous les fichiers SWF utilisant des objets partagés soient chargés uniquement dans une structure de pages HTML qui garantit que le fichier provient du même domaine que celui affiché dans la barre d'adresse du navigateur. Si vous essayez d'utiliser des objets partagés à partir d'un fichier SWF tiers et que l'utilisation des objets partagés tiers est désactivée, les méthodes `SharedObject.getLocal()` et `SharedObject.getRemote()` renvoient la valeur `null`. Pour plus d'informations, voir les adresses www.adobe.com/fr/products/flashplayer/articles/thirdpartylso.

Accès à la caméra, au microphone, au Presse-papiers, à la souris et au clavier

Flash Player 9 et les versions ultérieures, Adobe AIR 1.0 et les versions ultérieures

Lorsqu'un fichier SWF essaie d'accéder à la caméra ou au microphone de l'utilisateur à l'aide de la méthode `Camera.get()` ou `Microphone.get()`, Flash Player affiche une boîte de dialogue de confidentialité, dans laquelle l'utilisateur peut autoriser ou refuser l'accès à sa caméra ou son microphone. L'utilisateur et l'administrateur peuvent également désactiver l'accès à la caméra de manière globale ou pour chaque site, grâce aux commandes du fichier mms.cfg, de l'interface de paramétrage et du Gestionnaire des paramètres (voir « Contrôles administrateur » à la page 1092 et « Contrôles utilisateur » à la page 1093). Si des restrictions utilisateur s'appliquent, les méthodes `Camera.get()` et `Microphone.get()` renvoient chacune la valeur `null`. La propriété `Capabilities.avHardwareDisable` vous permet de déterminer si l'administrateur a interdit (valeur `true`) ou autorisé (valeur `false`) la caméra et le microphone.

La méthode `System.setClipboard()` autorise un fichier SWF à remplacer le contenu du Presse-papiers par une chaîne de caractères en texte brut, ce qui ne pose aucun risque de sécurité. Pour éviter que les mots de passe et autres données sensibles ne soient coupés ou copiés dans le Presse-papiers, il n'existe pas de méthode `getClipboard()` correspondante.

Une application qui s'exécute dans Flash Player peut uniquement contrôler les événements de clavier et de souris qui se produisent dans son focus. un contenu qui s'exécute dans Flash Player ne peut pas détecter d'événements de clavier ou de souris dans une autre application.

Sécurité AIR

Adobe AIR 1.0 et les versions ultérieures

Principes de sécurité AIR

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR s'exécutent dotées des mêmes restrictions de sécurité que les applications natives. En règle générale, les applications AIR, à l'instar des applications natives, disposent d'un accès étendu aux fonctionnalités du système d'exploitation, telles que la lecture et l'écriture de fichiers, le démarrage d'applications, les dessins sur l'écran et la communication avec le réseau. Les restrictions du système d'exploitation qui se rapportent aux applications natives, tels que les privilèges spécifiques aux utilisateurs, se rapportent également aux applications AIR.

Bien que le modèle de sécurité d'Adobe® AIR® constitue une évolution par rapport à celui d'Adobe® Flash® Player, le contrat de sécurité est différent de celui appliqué au contenu d'un navigateur. Ce contrat offre aux développeurs des moyens sûrs pour accéder à des fonctions beaucoup plus importantes et se lancer dans des expériences beaucoup plus enrichissantes et variées qui ne conviendraient pas à une application basée sur un navigateur.

Les applications AIR sont rédigées à l'aide de pseudo-code binaire compilé (contenu SWF) ou de script interprété (JavaScript, HTML) de sorte que le moteur d'exécution fournisse la gestion de la mémoire. Ceci minimise les risques que les applications AIR soient affectées par des vulnérabilités liées à la gestion de la mémoire, comme une surcharge de mémoire tampon ou une détérioration de mémoire. Voilà quelques-unes des vulnérabilités les plus courantes qui affectent les applications d'ordinateur de bureau rédigées en code natif.

Installation et mises à jour

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR sont distribuées par le biais des fichiers du programme d'installation AIR, dotés de l'extension `air` ou via des programmes d'installation natifs, dotés du format de fichier et de l'extension de la plate-forme native. Ainsi, le format du programme d'installation natif de Windows correspond à EXE, tandis que le format natif d'Android est APK.

Lorsqu'Adobe AIR est installé et qu'un fichier d'installation AIR est ouvert, le moteur d'exécution d'AIR gère le processus d'installation. Si un programme d'installation natif est utilisé, c'est le système d'exploitation qui gère le processus d'installation.

Remarque : *les développeurs peuvent spécifier une version, un nom d'application et une source d'éditeur lors de l'utilisation du format de fichier AIR, mais le flux initial de l'installation de l'application lui-même ne peut pas être modifié. Cette restriction est tout à l'avantage des utilisateurs car toutes les applications AIR partagent une procédure d'installation sûre, continue et stable, gérée par le moteur d'exécution. Si une personnalisation de l'application s'avère nécessaire, cela peut se faire lors de sa première exécution.*

Emplacement d'installation du moteur d'exécution

Adobe AIR 1.0 et les versions ultérieures

Pour les applications AIR qui font appel au format de fichier AIR, il est nécessaire d'installer préalablement le moteur d'exécution sur l'ordinateur d'un utilisateur, de même qu'il est nécessaire d'installer en premier lieu le module d'extension du navigateur Flash Player pour les fichiers SWF.

Le moteur d'exécution est installé à l'emplacement suivant sur un ordinateur de bureau :

- Mac OS : `/Bibliothèque/Frameworks/`
- Windows : `C:\Program Files\Common Files\Adobe AIR`
- Linux : `/opt/Adobe AIR/`

Sous Mac OS, pour installer une version mise à jour d'une application, l'utilisateur doit disposer de privilèges système appropriés pour accéder au répertoire de l'application. S'il utilise Windows ou Linux, l'utilisateur doit disposer de privilèges d'administrateur.

Remarque : *sous iOS, le moteur d'exécution d'AIR n'est pas installé séparément, puisque chaque application AIR est autonome.*

Le moteur d'exécution peut être installé de deux façons : via la fonction d'installation transparente (installation directe à partir d'un navigateur Web) ou via une installation manuelle. Les applications AIR mises en package sous forme de programmes d'installation natifs peuvent également installer le moteur d'exécution AIR dans le cadre de leur processus d'installation d'application normal. (Ce type de distribution du moteur d'exécution AIR requiert un accord de redistribution avec Adobe.)

Installation transparente (moteur d'exécution et application)

Adobe AIR 1.0 et les versions ultérieures

La fonction d'installation transparente fournit aux développeurs une expérience d'installation continue pour les utilisateurs qui n'ont pas encore Adobe AIR sur leur ordinateur. Dans la méthode d'installation transparente, le développeur crée un fichier SWF qui présente l'application à installer. Lorsqu'un utilisateur clique sur le fichier, celui-ci tente de détecter le moteur d'exécution. S'il n'est pas détecté, il est installé et immédiatement activé avec le processus d'installation pour l'application du développeur.

Installation manuelle

Adobe AIR 1.0 et les versions ultérieures

D'un autre côté, l'utilisateur peut manuellement télécharger et installer le moteur d'exécution avant d'ouvrir un fichier AIR. Le développeur peut alors distribuer un fichier AIR de différentes façons : par exemple, par e-mail ou un lien HTML sur un site Web. Lorsque le fichier AIR est ouvert, le moteur d'exécution se lance dans l'installation de l'application.

Flux d'installation de l'application

Adobe AIR 1.0 et les versions ultérieures

Le modèle de sécurité AIR permet aux utilisateurs de décider s'il y a lieu d'installer une application AIR. L'expérience dans l'installation d'AIR a permis d'apporter plusieurs améliorations par rapport aux technologies d'installation des applications natives, ce qui rend cette décision de confiance plus facile à prendre pour les utilisateurs :

- Le moteur d'exécution fournit une expérience d'installation stable sur tous les systèmes d'exploitation, même lorsqu'une application AIR est installée à partir d'un lien dans un navigateur Web. La plupart des expériences d'installation des applications dépendent d'un navigateur ou d'une autre application pour fournir des informations sur la sécurité, si encore elles existent.
- L'expérience d'installation de l'application AIR identifie la source de l'application et les informations sur la nature des privilèges disponibles pour l'application, à condition que l'utilisateur permette à l'installation de se poursuivre.
- Le moteur d'exécution gère le processus d'installation d'une application AIR. Une application AIR n'est pas en mesure d'intervenir dans le processus d'installation que le moteur d'exécution utilise.

En règle générale, les utilisateurs ne devraient pas installer d'application d'ordinateur de bureau qui provient d'une source à laquelle ils ne font pas confiance ou qui ne peut pas être authentifiée. Le fardeau de la preuve sur la sécurité pour les applications natives est également valable pour les applications AIR comme elle l'est pour d'autres applications à installer.

Destination de l'application

Adobe AIR 1.0 et les versions ultérieures

Le répertoire d'installation peut être défini à l'aide de l'une des deux options suivantes :

1. L'utilisateur personnalise la destination au cours de l'installation. L'application s'installe à l'emplacement que lui indique l'utilisateur.
2. Si celui-ci ne modifie pas la destination de l'installation, l'application s'installe suivant le chemin par défaut fourni par le moteur d'exécution :
   - Mac OS : `~/Applications/`

- Windows XP (et versions antérieures) : `C:\Program Files\`
- Windows Vista : `~/Apps/`
- Linux : /opt/

Si le développeur spécifie un paramètre `installFolder` dans le fichier descripteur d'application, l'application s'installe dans un sous-répertoire de ce dossier.

Système de fichiers AIR

Adobe AIR 1.0 et les versions ultérieures

Le processus d'installation pour les applications AIR copie tous les fichiers que le développeur a inclus au sein du fichier d'installation d'AIR dans l'ordinateur local de l'utilisateur. L'application installée est composée des éléments suivants :

- Windows : un répertoire qui contient tous les fichiers inclus dans le fichier d'installation d'AIR. Le moteur d'exécution crée aussi un fichier exe au cours de l'installation de l'application AIR.
- Linux : un répertoire qui contient tous les fichiers inclus dans le fichier d'installation d'AIR. Le moteur d'exécution crée également un fichier bin au cours de l'installation de l'application AIR.
- Mac OS : un fichier `app` qui reprend tout le contenu du fichier d'installation d'AIR. Il peut être inspecté à l'aide de l'option « Afficher le contenu du package » dans Finder. Le moteur d'exécution crée ce fichier app dans le cadre de l'installation de l'application AIR.

Une application AIR est exécutée par :

- Windows : l'exécution du fichier .exe dans le dossier install ou un raccourci qui correspond à ce fichier (comme un raccourci dans le menu Démarrer ou sur le Bureau).
- Linux : l'exécution du fichier .bin dans le dossier install, la sélection de l'application dans le menu Applications ou l'exécution d'un alias ou d'un raccourci.
- Mac OS : l'exécution du fichier .app ou un alias qui pointe dessus.

Le système des fichiers d'application contient aussi des sous-répertoires liés à la fonction de l'application. Par exemple, les informations envoyées à la mémoire locale chiffrée sont enregistrées dans un sous-répertoire qui porte le même nom que l'identifiant d'application de l'application.

Mémoire de l'application AIR

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR disposent de privilèges pour écrire dans n'importe quel emplacement du disque dur de l'utilisateur ; toutefois, il est recommandé aux développeurs d'utiliser le chemin `app-storage:/` pour le stockage local lié à leur application. Les fichiers écrits dans `app-storage:/` à partir d'une application se trouvent à un emplacement standard qui dépend de leur système d'exploitation :

- Sous Mac OS : le répertoire de stockage d'une application varie selon la version d'AIR :
  - **AIR 3.2 et les versions ultérieures** - `<appData>/<appId>/Local Store/` où `<appData>` correspond au « dossier des préférences » de l'utilisateur, c'est-à-dire en règle générale : `/Utilisateurs/<user>/Bibliothèque/Préférences`

- **AIR 3.3 et les versions ultérieures** - `<chemin>/Bibliothèque/Application Support/<appID>/Local Store`, où `<chemin>` est soit `/Utilisateurs/<utilisateur>/Bibliothèque/Containers/<bundle-id>/Data` (environnement de sandbox), soit `/Utilisateurs/<utilisateur>` (exécution en dehors d'un environnement de sandbox)

- Sous Windows : le répertoire de stockage d'une application se nomme `<appData>\<appId>\Local Store\` , où `<appData>` correspond au « dossier spécial » CSIDL_APPDATA de l'utilisateur, le plus souvent `C:\Documents and Settings\<nom d'utilisateur>\Application Data`
- Sous Linux : `<appData>/<appID>/Local Store/`, où `<appData>` correspond à `/home/<utilisateur>/.appdata`

Vous pouvez accéder au répertoire de stockage de l'application via la propriété `air.File.applicationStorageDirectory`. Vous pouvez accéder à son contenu à l'aide la méthode `resolvePath()` de la classe Fichier. Pour plus d'informations, voir la section « Utilisation du système de fichiers » à la page 676.

Mise à jour d'Adobe AIR

Adobe AIR 1.0 et les versions ultérieures

Lorsque l'utilisateur installe une application AIR qui nécessite une version mise à jour du moteur d'exécution, celui-ci installe automatiquement la version appropriée.

Pour mettre à jour le moteur d'exécution, un utilisateur doit disposer des privilèges administrateur pour l'ordinateur.

Mise à jour des applications AIR

Adobe AIR 1.0 et les versions ultérieures

Le développement et le déploiement des mises à jour des logiciels constituent l'un des défis les plus importants dans le domaine de la sécurité auxquels sont confrontées les applications rédigées en code natif. L'interface de programmation AIR fournit un mécanisme pour améliorer cette situation : la méthode `Updater.update()` peut être appelée lors du lancement afin de vérifier un emplacement distant pour un fichier AIR. Si une mise à jour s'impose, le fichier AIR est téléchargé, installé et l'application redémarre. Les développeurs peuvent utiliser cette classe, non seulement pour obtenir de nouvelles fonctions, mais également pour réagir à des risques de sécurité potentiels.

La classe Updater est réservée à la mise à jour des applications distribuées sous forme de fichiers AIR. Les applications distribuées en tant qu'applications natives doivent utiliser les fonctions de mise à jour éventuellement intégrées au système d'exploitation natif.

Remarque : *les développeurs peuvent spécifier la version d'une application en paramétrant la propriété versionNumber du fichier descripteur d'application.*

Désinstallation d'une application AIR

Adobe AIR 1.0 et les versions ultérieures

La suppression d'une application AIR entraîne celle de tous les fichiers de son répertoire. Elle ne supprime toutefois pas tous les fichiers que l'application a éventuellement écrits hors de ce répertoire. Les changements apportés à des fichiers par l'application AIR hors du répertoire de l'application ne sont pas affectés par une suppression des applications AIR.

Paramètres du Registre Windows pour les administrateurs

Adobe AIR 1.0 et les versions ultérieures

Sous Windows, les administrateurs peuvent configurer une machine pour empêcher (ou permettre) les mises à jour de l'installation de l'application AIR et du moteur d'exécution. Ces paramètres se trouvent dans le Registre Windows sous la clé suivante : HKLM\Software\Policies\Adobe\AIR. Parmi eux figurent :

Paramètre du Registre	Description
AppInstallDisabled	Indique que l'installation et la désinstallation de l'application AIR sont autorisées. Définissez sur 0 pour « autorisé » et 1 pour « interdit ».
UntrustedAppInstallDisabled	Indique que l'installation d'applications AIR non fiables (applications qui ne possèdent pas de certificat fiable) est autorisée. Définissez sur 0 pour « autorisé » et 1 pour « interdit ».
UpdateDisabled	Indique que la mise à jour du moteur d'exécution est autorisée, soit comme tâche de fond, soit comme faisant partie d'une installation explicite. Définissez sur 0 pour « autorisé » et 1 pour « interdit ».

Sécurité HTML dans Adobe AIR

Adobe AIR 1.0 et les versions ultérieures

Cette section décrit l'architecture de la sécurité HTML dans AIR, ainsi que l'utilisation d'iframes, d'images et du pont de sandbox pour configurer les applications HTML et sécuriser l'intégration de contenu HTML à des applications SWF.

Le moteur d'exécution applique des règles et fournit des mécanismes pour compenser des vulnérabilités de sécurité possibles dans HTML et JavaScript. Les mêmes règles s'appliquent que votre application soit rédigée principalement en JavaScript ou que vous chargiez le contenu HTML et JavaScript dans une application basée sur SWF. Les contenus du sandbox de l'application et du sandbox de sécurité hors application possèdent des privilèges différents. Lorsque vous chargez un contenu dans une iframe ou une image, le moteur d'exécution fournit un mécanisme *pont de sandbox* sécurisé qui permet au contenu de l'image ou de l'iframe de communiquer de façon sûre avec le contenu du sandbox de sécurité de l'application.

Le SDK d'AIR intègre trois classes de rendu du contenu HTML.

La classe HTMLLoader assure une intégration étroite entre le code JavaScript et les API d'AIR.

La classe StageWebView est une classe de rendu HTML et son intégration à l'application AIR hôte est extrêmement réduite. Le contenu chargé par la classe StageWebView n'est jamais placé dans le sandbox de sécurité de l'application et il lui est impossible d'accéder aux données ou d'appeler des fonctions dans l'application AIR hôte. Sur les plates-formes de poste de travail, la classe StageWebView fait appel au moteur HTML AIR intégré, basé sur Webkit, qui est également utilisé par la classe HTMLLoader. Sur les plates-formes mobiles, elle fait appel au contrôle HTML fourni par le système d'exploitation. Par conséquent, sur les plates-formes mobiles, la classe StageWebView est soumise aux mêmes considérations de sécurité et souffre des mêmes vulnérabilités que le navigateur Web du système.

La classe TextField peut afficher des chaînes de texte HTML. Il est impossible d'exécuter du code JavaScript, mais le texte peut inclure des liens et des images chargées en externe.

Pour plus d'informations, voir la section « Contournement des erreurs JavaScript liées à la sécurité » à la page 1021.

Aperçu de la configuration de vos applications à base d'HTML

Adobe AIR 1.0 et les versions ultérieures

Les images et iframes fournissent une structure adéquate pour l'organisation du contenu HTML dans AIR. Les images fournissent des moyens qui permettent à la fois de maintenir la persistance des données et pour travailler de façon sécurisée avec du contenu distant.

Comme HTML dans AIR conserve son organisation normale, basée sur des pages, l'environnement HTML est intégralement actualisé si l'image supérieure de votre contenu HTML « navigue » dans une page différente. Vous pouvez utiliser des images et des iframes pour maintenir la persistance des données dans AIR pratiquement de la même façon que vous le feriez pour une application Web qui s'exécuterait dans un navigateur. Définissez les principaux objets de votre application dans l'image supérieure et ils persisteront aussi longtemps que vous n'autoriserez pas l'image à naviguer dans une nouvelle page. Utilisez des images ou des iframes enfant pour charger et afficher des parties transitoires de l'application. Il existe plusieurs façons de maintenir la persistance des données, en plus des images ou bien à leur place. Par exemple, des cookies, des objets locaux partagés, l'emplacement de stockage des fichiers locaux, celui des fichiers chiffrés et celui des bases de données locales.

Comme HTML dans AIR conserve sa frontière normale et imprécise entre le code exécutable et les données, AIR place le contenu de l'image supérieure de l'environnement HTML dans le sandbox de l'application. Après l'événement `load` de la page, AIR restreint toutes les opérations, telles que `eval()`, susceptibles de convertir une chaîne de texte en un objet exécutable. Cette restriction est imposée même lorsqu'une application ne charge pas du contenu distant. Pour que du contenu HTML distant puissent exécuter ces opérations restreintes, vous devez utiliser des images ou des iframes pour placer le contenu dans un sandbox hors application. (L'exécution de contenu dans une image enfant affectée à un sandbox peut s'avérer nécessaire lorsque vous utilisez des structures d'application JavaScript qui reposent sur la fonction `eval()`.) Pour un répertoire complet des restrictions relatives à JavaScript dans le sandbox de l'application, voir la section « Restrictions relatives au code pour un contenu dans des sandbox différents » à la page 1130.

Comme HTML dans AIR conserve sa capacité à charger du contenu distant et possiblement non sécurisé, AIR applique une politique de « même origine » qui empêche le contenu d'un domaine d'interagir avec celui de l'autre. Pour permettre une interaction entre contenu de l'application et contenu d'un autre domaine, vous pouvez configurer un pont qui servira d'interface entre une image parent et une image enfant.

Configuration d'un rapport de sandbox parent-enfant

Adobe AIR 1.0 et les versions ultérieures

AIR ajoute les attributs `sandboxRoot` et `documentRoot` aux éléments image et iframe de HTML. Ces attributs vous permettent de traiter le contenu de l'application comme s'il provenait d'un autre domaine :

Attribut	Description
sandboxRoot	L'URL à utiliser pour déterminer le sandbox et le domaine dans lesquels placer le contenu de l'image. Les modèles d'URL `file:`, `http:` ou `https:` doivent être utilisés.
documentRoot	L'URL à partir de laquelle il faut charger le contenu de l'image. Les modèles d'URL `app:`, `http:` ou `app-storage:` doivent être utilisés.

L'exemple ci-dessous mappe le contenu installé dans le sous-répertoire du sandbox de l'application qui doit s'exécuter dans le sandbox distant et le domaine www.example.com :

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Configuration d'un pont entre images parent et enfant dans des sandbox ou des domaines différents

Adobe AIR 1.0 et les versions ultérieures

AIR ajoute les propriétés `childSandboxBridge` et `parentSandboxBridge` à l'objet `window` de toute image enfant. Ces propriétés vous permettent de définir des ponts qui serviront d'interfaces entre une image parent et une image enfant. Chaque pont ne va que dans une seule direction :

`childSandboxBridge` : la propriété `childSandboxBridge` permet à l'image enfant de présenter une interface au contenu dans l'image parent. Pour présenter une interface, vous définissez la propriété `childSandbox` sur une fonction ou un objet dans l'image enfant. Vous pouvez alors accéder à l'objet ou à la fonction à partir du contenu dans l'image parent. L'exemple ci-dessous montre comment un script qui s'exécute dans une image enfant peut présenter un objet contenant une fonction ou une propriété à son parent :

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Si ce contenu enfant est dans une iframe affectée d'une `id child`, vous pouvez accéder à l'interface à partir du contenu parent en lisant la propriété `childSandboxBridge` de l'image :

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

`parentSandboxBridge` : la propriété `parentSandboxBridge` permet à l'image parent de présenter une interface au contenu dans l'image enfant. Pour présenter une interface, vous définissez la propriété `parentSandbox` de l'image enfant sur une fonction ou un object dans l'image parent. Vous pouvez alors accéder à l'objet ou à la fonction à partir du contenu dans l'image enfant. L'exemple ci-dessous montre comment un script qui s'exécute dans l'image parent peut présenter un objet contenant une fonction save à un enfant :

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

Avec cette interface, le contenu de l'image enfant pourrait enregistrer du texte dans un fichier appelé save.txt. Toutefois, il ne pourrait avoir aucun autre accès au système de fichiers. En règle générale, le contenu de l'application devrait présenter l'interface la plus étroite possible aux autres sandbox. Le contenu enfant pourrait appeler la fonction save comme suit :

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Si le contenu enfant tente de définir une propriété de l'objet`parentSandboxBridge`, le moteur d'exécution renvoie une exception SecurityError. Si le contenu parent tente de définir une propriété de l'objet`childSandboxBridge`, le moteur d'exécution renvoie une exception SecurityError.

Restrictions relatives au code pour un contenu dans des sandbox différents

Adobe AIR 1.0 et les versions ultérieures

Comme nous l'avons vu dans l'introduction à cette rubrique, « Sécurité HTML dans Adobe AIR » à la page 1127, le moteur d'exécution applique les règles et fournit des mécanismes pour compenser les vulnérabilités de sécurité possibles dans HTML et JavaScript. La présente rubrique répertorie ces restrictions. Si le code tente d'appeler ces interfaces de programmation réservées, le moteur d'exécution renvoie une erreur accompagnée du message « Violation des règles de sécurité dans le sandbox de sécurité de l'application par le moteur d'exécution d'Adobe AIR sur du code JavaScript ».

Pour plus d'informations, voir la section « Contournement des erreurs JavaScript liées à la sécurité » à la page 1021.

Restrictions relatives à l'utilisation de la fonction eval() de JavaScript et de techniques similaires

Adobe AIR 1.0 et les versions ultérieures

Pour le contenu HTML dans le sandbox de sécurité de l'application, il existe des limites dans l'utilisation des interfaces de programmation susceptibles de transformer dynamiquement les chaînes en code exécutable après le chargement de ce code (après que l'événement `onload` de l'élément `body` a été distribué et que l'exécution de la fonction du gestionnaire `onload` est terminée). Ceci empêche l'application d'injecter (et d'exécuter) du code par inadvertance à partir de sources non-applicatives, comme des domaines du réseau potentiellement non sécurisés.

Par exemple, si votre application utilise des données de chaîne à partir d'une source distante pour écrire dans la propriété innerHTML d'un élément DOM, la chaîne pourrait inclure du code exécutable (JavaScript) et susceptible d'exécuter des opérations non sécurisées. Toutefois, tant que le contenu est en cours de chargement, il n'y a pas de risque que des chaînes distantes soient insérées dans le DOM.

L'utilisation de la fonction `eval()` de JavaScript constitue une restriction. Dès que le code du sandbox de l'application est chargé et après le traitement du gestionnaire d'événement onload, vous ne pouvez utiliser la fonction `eval()` que dans certaines conditions. Les règles suivantes s'appliquent dans l'utilisation de la fonction `eval()` *après* le chargement du code à partir du sandbox de sécurité de l'application :

- Les expressions contenant des littéraux sont autorisées. Exemple :

```
eval("null");
eval("3 + .14");
eval("'foo'");
```

- Les littéraux d'objet sont autorisés, comme dans les cas décrits ci-dessous :

```
{ prop1: val1, prop2: val2 }
```

- Les littéraux d'objet de lecture et de définition (getter/setter) sont *interdits* comme dans les exemples ci-dessous :

```
{ get prop1() { ... }, set prop1(v) { ... } }
```

- Les littéraux de tableau sont autorisés, comme dans les cas décrits ci-dessous :

```
[ val1, val2, val3 ]
```

- Les expressions impliquant des lectures de propriété sont *interdites* comme dans ce qui suit :

```
a.b.c
```

- Le lancement de fonctions est *interdit*.
- Les définitions de fonctions sont *interdites*.
- Le paramétrage de toute propriété est *interdit*.
- Les littéraux de fonctions sont *interdits*.

Toutefois, au cours du chargement du code, avant l'événement `onload` et au cours de l'exécution de la fonction du gestionnaire d'événement `onload`, ces restrictions ne s'appliquent pas au contenu du sandbox de sécurité de l'application.

Par exemple, après le chargement du code, l'exécution du code ci-dessous dans le moteur d'exécution renvoie une exception.

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

Un code généré dynamiquement, comme celui qui est produit par l'appel de la fonction `eval()`, présenterait un risque pour la sécurité s'il était autorisé au sein du sandbox de l'application. Par exemple, une application peut exécuter par inadvertance une chaîne chargée à partir d'un domaine de réseau et celle-ci peut contenir du code malveillant. Par exemple, il peut s'agir de code conçu pour la suppression ou la modification de fichiers dans l'ordinateur de l'utilisateur. Ou bien encore de code qui renvoie à un domaine de réseau non approuvé une copie du contenu d'un fichier local.

Vous trouverez ci-dessous des façons de générer du code dynamique :

- Appel de la fonction `eval()`.
- Utilisation des propriétés `innerHTML` ou des fonctions DOM pour insérer des balises de script qui chargent un script hors du répertoire de l'application.
- Utilisation des propriétés `innerHTML` ou des fonctions DOM pour insérer des balises de script qui possèdent du code incorporé (plutôt que de charger un script au moyen de l'attribut `src`).
- Paramétrage de l'attribut `src` pour qu'une balise `script` charge un fichier JavaScript qui est hors du répertoire de l'application.
- Utilisation du modèle d'URL de `JavaScript` comme dans (`href="javascript:alert('Test')"`).
- Utilisation de la fonction `setInterval()` ou `setTimout()`, où le premier paramètre (qui définit la fonction pour qu'elle s'exécute de façon asynchrone) est une chaîne (à évaluer) plutôt qu'un nom de fonction (comme dans `setTimeout('x = 4', 1000)`).
- Appel de `document.write()` ou de `document.writeln()`.

Le code dans le sandbox de sécurité de l'application ne peut utiliser ces méthodes au cours du chargement du contenu.

Ces restrictions n'empêchent *pas* l'utilisation d'`eval()` avec des littéraux d'objet JSON. Ceci permet au contenu de votre application de travailler avec la bibliothèque JavaScript de JSON. Il vous est néanmoins interdit d'utiliser du code JSON surchargé (avec des gestionnaires d'événement).

Pour d'autres structures Ajax et bibliothèques de code JavaScript, assurez-vous que le code de la structure ou de la bibliothèque fonctionne sur du code généré dynamiquement, dans le cadre de ces restrictions. Si ce n'est pas le cas, placez tout contenu qui utilise la structure ou la bibliothèque dans un sandbox de sécurité hors application. Pour plus d'informations, voir « Restrictions associées au contenu JavaScript dans AIR » à la page 1089 et « Programmation entre contenu d'application et contenu hors application » à la page 1139. Adobe maintient un répertoire de structures Ajax, reconnues pour leur prise en charge du sandbox de sécurité de l'application, à l'adresse http://www.adobe.com/products/air/develop/ajax/features/.

Contrairement au contenu du sandbox de sécurité de l'application, celui de JavaScript dans un sandbox de sécurité hors application *peut* à tout moment appeler la fonction `eval()` pour exécuter dynamiquement le code généré.

Restrictions relatives à l'accès aux interfaces de programmation AIR (pour des sandbox hors application)

Adobe AIR 1.0 et les versions ultérieures

Le code JavaScript dans un sandbox hors application n'a pas accès à l'objet `window.runtime` et, de ce fait, il ne peut pas exécuter d'interfaces de programmation AIR. Si un contenu, dans un sandbox de sécurité hors application, appelle le code ci-dessous, l'application renvoie une exception TypeError.

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

Le type d'exception est TypeError (valeur non définie) parce que le contenu du sandbox hors application ne reconnaît pas l'objet `window.runtime` de sorte qu'il est considéré comme une valeur non définie.

Vous pouvez présenter les fonctionnalités du moteur d'exécution dans un sandbox hors application à l'aide d'un pont de script. Pour plus d'informations, voir « Programmation entre contenu d'application et contenu hors application » à la page 1139.

Restrictions relatives à l'utilisation des appels XMLHttpRequest

Adobe AIR 1.0 et les versions ultérieures

Le contenu HTML du sandbox de sécurité de l'application utilise des méthodes XMLHttpRequest pour charger des données à partir d'emplacements hors du sandbox de l'application au cours du chargement du contenu HTML et de l'exécution de l'événement `onLoad`.

Par défaut, un contenu HTML dans des sandbox de sécurité hors application n'est pas autorisé à utiliser l'objet XMLHttpRequest de JavaScript pour charger des données à partir des domaines autres que le domaine qui appelle la requête. Une balise `image` ou `iframe` peut contenir un attribut `allowcrosscomainxhr`. La définition de cet attribut sur toute valeur non nulle permet au contenu de l'image ou de l'iframe d'utiliser l'objet XMLHttpRequest de JavaScript pour charger les données à partir de domaines autres que celui du code qui appelle la requête.

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

Pour plus d'informations, voir « Programmation entre contenus de différents domaines » à la page 1133.

Restrictions relatives au chargement d'éléments de CSS, d'image, d'iframe et d'img (pour un contenu dans des sandbox hors application)

Adobe AIR 1.0 et les versions ultérieures

Un contenu HTML dans un sandbox de sécurité distant (réseau) ne peut charger que du contenu CSS, `image`, `iframe` et `img` à partir de sandbox distants (à partir d'URL du réseau).

Un contenu HTML dans un sandbox local avec système de fichiers, local avec réseau ou approuvé localement ne peut charger que du contenu CSS, image, iframe et `img` à partir de sandbox locaux (mais pas à partir de sandbox d'application ou distants).

Restrictions relatives à l'appel de la méthode window.open() de JavaScript

Adobe AIR 1.0 et les versions ultérieures

Si une fenêtre créée par un appel à la méthode `window.open()` de JavaScript affiche un contenu à partir d'un sandbox de sécurité hors application, le titre de la fenêtre commence par celui de la fenêtre principale (de lancement), suivi du caractère deux points (:). Il n'est pas possible d'utiliser du code pour retirer cette partie du titre de l'écran.

Le contenu d'un sandbox de sécurité hors application ne peut réussir un appel à la méthode `window.open()` de JavaScript que s'il répond à un événement déclenché par la souris ou le clavier de l'utilisateur. Cette condition permet d'éviter qu'un contenu hors application ne crée des fenêtres qui pourraient être utilisées de façon trompeuse, par exemple pour des attaques d'hameçonnage. En outre, le gestionnaire d'événement pour l'événement de la souris ou du clavier ne peut pas paramétrer la méthode `window.open()` pour qu'elle s'exécute au bout d'un certain temps, par exemple en appelant la fonction `setTimeout()`.

Le contenu d'un sandbox distant (réseau) ne peut utiliser la méthode `window.open()` que pour ouvrir le contenu dans un sandbox de réseau distant. Il ne peut pas utiliser cette méthode pour ouvrir du contenu à partir de l'application ou d'un sandbox local.

Un contenu de sandbox local avec système de fichiers, sandbox local avec réseau ou sandbox approuvé localement (voir « Sandbox de sécurité » à la page 1087) ne fait appel à la méthode `window.open()` que pour ouvrir un contenu dans un sandbox local. Il ne peut pas utiliser cette méthode pour ouvrir du contenu à partir de l'application ou d'un sandbox distant.

Erreurs lors de l'appel de code interdit

Adobe AIR 1.0 et les versions ultérieures

Si vous appelez du code interdit d'exécution dans un sandbox en raison de ces restrictions liées à la sécurité, le moteur d'exécution distribue une erreur JavaScript : « Violation des règles de sécurité dans le sandbox de sécurité de l'application par le moteur d'exécution d'Adobe AIR sur du code JavaScript ».

Pour plus d'informations, voir la section « Contournement des erreurs JavaScript liées à la sécurité » à la page 1021.

Protection du sandbox lors du chargement de contenu HTML depuis une chaîne

Adobe AIR 1.0 et les versions ultérieures

La méthode `loadString()` de la classe HTMLLoader vous permet de créer du contenu HTML au moment de l'exécution. Toutefois, les données utilisées comme contenu HTML peuvent être corrompues lorsqu'elles sont chargées à partir d'une source Internet non sécurisée. De ce fait, par défaut, le contenu HTML créé à l'aide de la méthode `loadString()` n'est pas placé dans le sandbox de l'application et n'a pas accès aux API AIR. Vous pouvez cependant définir la propriété `placeLoadStringContentInApplicationSandbox` d'un objet HTMLLoader sur true pour placer le contenu HTML créé avec la méthode `loadString()` dans le sandbox de l'application. Pour plus d'informations, voir la section « Chargement de contenu HTML depuis une chaîne » à la page 1020.

Programmation entre contenus de différents domaines

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR disposent de privilèges spéciaux lorsqu'elles sont installées. Il est essentiel que les mêmes privilèges ne soient pas dévoilés à un autre contenu, y compris les fichiers distants et les fichiers locaux qui ne font pas partie de l'application.

A propos du pont de sandbox AIR

Adobe AIR 1.0 et les versions ultérieures

Généralement, le contenu d'un autre domaine ne peut pas appeler de scripts d'autres domaines. Pour protéger les applications AIR d'une divulgation accidentelle d'informations ou de contrôles privilégiés, les restrictions suivantes sont placées dans le contenu du sandbox de sécurité de `l'application` (contenu installé avec l'application) :

- Le code dans le sandbox de sécurité de l'application ne peut pas autoriser l'accès aux autres sandbox par un appel à la méthode `Security.allowDomain()`. L'appel à cette méthode depuis le sandbox de sécurité de l'application renvoie une erreur.
- Il est impossible d'importer un contenu hors application dans un sandbox de l'application en paramétrant la propriété `LoaderContext.securityDomain` ou `LoaderContext.applicationDomain`.

Il subsiste encore des situations où l'application AIR principale demande un contenu à partir du domaine distant pour avoir un accès contrôlé à des scripts de celle-ci, ou vice-versa. Pour ce faire, le moteur d'exécution fournit un mécanisme de *pont de sandbox* qui sert de passerelle entre deux sandbox. Un pont de sandbox peut fournir une interaction explicite entre sandbox de sécurité distant et d'application.

Le pont de sandbox présente deux objets auxquels les scripts chargés et en cours de chargement peuvent tous deux accéder :

- L'objet `parentSandboxBridge` permet au contenu en cours de chargement de présenter des propriétés et des fonctions à des scripts du contenu chargé.
- L'objet `childSandboxBridge` permet au contenu chargé de présenter des propriétés et des fonctions à des scripts dans le contenu en cours de chargement.

Ce sont les valeurs et non les références qui sont transmises par le pont de sandbox. Toutes les données sont sérialisées. Ce qui signifie que les objets présentés par l'un des côtés du pont ne peuvent pas être définis par l'autre et qu'aucun des objets présentés n'est typé. De plus, vous ne pouvez présenter que des objets et des fonctions simples (pas complexes).

Si le contenu enfant tente de définir une propriété de l'objetparentSandboxBridge, le moteur d'exécution renvoie une exception SecurityError. De même, si le contenu parent tente de définir une propriété de l'objetchildSandboxBridge, le moteur d'exécution renvoie une exception SecurityError.

Exemple de pont de sandbox (SWF)

Adobe AIR 1.0 et les versions ultérieures

Supposez qu'une application de disquaire AIR souhaite autoriser des fichiers distants à diffuser le prix des albums, mais qu'il ne souhaite pas que le fichier SWF distant dévoile s'il s'agit d'un prix de vente. Pour ce faire, une classe StoreAPI fournit une méthode d'acquisition de prix tout en masquant le prix de vente. Une occurrence de cette classe StoreAPI est alors affectée à la propriété `parentSandboxBridge` de l'objet LoaderInfo dont l'objet Loader charge le fichier SWF distant.

Voici le code pour le disquaire AIR :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
title="Music Store" creationComplete="initApp()">
    <mx:Script>
        import flash.display.Loader;
        import flash.net.URLRequest;

        private var child:Loader;
        private var isSale:Boolean = false;

        private function initApp():void {
            var request:URLRequest =
                    new URLRequest("http://[www.yourdomain.com]/PriceQuoter.swf")

            child = new Loader();
            child.contentLoaderInfo.parentSandboxBridge = new StoreAPI(this);
            child.load(request);
            container.addChild(child);
        }
        public function getRegularAlbumPrice():String {
            return "$11.99";
        }
        public function getSaleAlbumPrice():String {
            return "$9.99";
        }
        public function getAlbumPrice():String {
            if(isSale) {
                return getSaleAlbumPrice();
            }
            else {
                return getRegularAlbumPrice();
            }
        }
    </mx:Script>
    <mx:UIComponent id="container" />
</mx:WindowedApplication>
```

L'objet StoreAPI appelle l'application principale pour récupérer le prix courant de l'album mais renvoie « Non disponible » lorsque la méthode `getSaleAlbumPrice()` est appelée. Le code ci-dessous définit la classe StoreAPI :

```
public class StoreAPI
{
    private static var musicStore:Object;

    public function StoreAPI(musicStore:Object)
    {
        this.musicStore = musicStore;
    }

    public function getRegularAlbumPrice():String {
        return musicStore.getRegularAlbumPrice();
    }

    public function getSaleAlbumPrice():String {
        return "Not available";
    }

    public function getAlbumPrice():String {
        return musicStore.getRegularAlbumPrice();
    }
}
```

Le code suivant représente un exemple d'un fichier SWF PriceQuoter qui rend compte du prix du disquaire mais pas de celui de la vente.

```
package
{
    import flash.display.Sprite;
    import flash.system.Security;
    import flash.text.*;

    public class PriceQuoter extends Sprite
    {
        private var storeRequester:Object;

        public function PriceQuoter() {
            trace("Initializing child SWF");
            trace("Child sandbox: " + Security.sandboxType);
            storeRequester = loaderInfo.parentSandboxBridge;

            var tf:TextField = new TextField();
            tf.autoSize = TextFieldAutoSize.LEFT;
            addChild(tf);

            tf.appendText("Store price of album is: " + storeRequester.getAlbumPrice());
            tf.appendText("\n");
            tf.appendText("Sale price of album is: " + storeRequester.getSaleAlbumPrice());
        }
    }
}
```

Exemple de pont de sandbox (HTML)

Adobe AIR 1.0 et les versions ultérieures

Dans un contenu HTML, les propriétés `parentSandboxBridge` et `childSandboxBridge` sont ajoutées à l'objet Window de JavaScript d'un document enfant. Pour un exemple sur la façon de configurer des fonctions de pont dans un contenu HTML, voir la section « Configuration d'une interface de pont de sandbox » à la page 1040.

Limitation de l'exposition de l'interface de programmation

Adobe AIR 1.0 et les versions ultérieures

Lorsque vous présentez des ponts de sandbox, il est important de présenter des interfaces de programmation de haut niveau qui limitent le degré d'exposition à des abus. Nous vous rappelons que le contenu qui appelle votre implémentation de pont peut être en danger (par exemple, par une injection de code). Ainsi, par exemple, la présentation de la méthode `readFile(path:String)` (qui lit le contenu d'un fichier arbitraire) par un pont est exposé à un risque. Il serait préférable de présenter une interface de programmation `readApplicationSetting()` qui ne suit pas un chemin mais qui lit un fichier spécifique. L'approche plus sémantique limite les dommages qu'une application peut causer dès qu'une de ses parties est affectée.

Voir aussi

« Programmation croisée du contenu dans des sandbox de sécurité distincts » à la page 1038

« Sandbox de l'application AIR » à la page 1088

Ecriture sur un disque

Adobe AIR 1.0 et les versions ultérieures

Les applications qui s'exécutent dans un navigateur Web n'ont qu'une interaction limitée avec le système de fichiers local de l'utilisateur. Les navigateurs Web implémentent des stratégies de sécurité qui garantissent que l'ordinateur de l'utilisateur ne peut pas être mis en danger à la suite du chargement d'un contenu du Web. Par exemple, les fichiers SWF qui passent par Flash Player dans un navigateur ne peuvent pas interagir directement avec des fichiers qui se trouvent déjà sur l'ordinateur d'un utilisateur. Les objets partagés et les cookies peuvent être écrits dans l'ordinateur d'un utilisateur dans le but de maintenir les préférences et les données de l'utilisateur, mais il s'agit là de la limite de l'interaction avec le système de fichiers. Comme les applications AIR sont installées en mode natif, ils possèdent un contrat de sécurité différent, celui qui inclut la possibilité de lire et d'écrire dans tout le système de fichiers local.

Cette liberté s'accompagne d'une grande responsabilité de la part des développeurs. Les insécurités accidentelles de l'application compromettent non seulement la fonctionnalité de l'application, mais aussi l'intégrité de l'ordinateur de l'utilisateur. C'est pourquoi les développeurs devraient lire la section « Recommandations destinées aux développeurs en matière de sécurité » à la page 1140.

Les développeurs AIR peuvent accéder à des fichiers du système de fichiers local et en placer dans ce système à l'aide de plusieurs conventions de modèles d'URL :

Modèle d'URL	Description
app:/	Un alias du répertoire de l'application. Les fichiers auxquels on accède à partir de ce chemin sont affectés au sandbox de l'application et sont dotés du maximum de privilèges par le moteur d'exécution.
app-storage:/	Un alias du répertoire de stockage local de l'application, standardisé par le moteur d'exécution. Les fichiers auxquels on accède à partir de ce chemin sont affectés au sandbox hors application.
file:///	Un alias qui représente la racine du disque dur de l'utilisateur. Un fichier auquel on accède à partir de ce chemin est affecté à un sandbox de l'application si le fichier existe dans le répertoire de l'application et à un sandbox hors application dans les autres cas.

Remarque : *les applications AIR ne peuvent pas modifier le contenu à l'aide du modèle d'URL app:. De plus, le répertoire de l'application peut être lu uniquement en raison des paramètres de l'administrateur.*

A moins qu'il n'existe des restrictions de l'administrateur à propos de l'ordinateur de l'utilisateur, les applications AIR détiennent le privilège d'écrire sur tout emplacement du disque dur de l'utilisateur. Les développeurs sont informés d'utiliser le chemin `app-storage:/` pour le stockage local lié à leur application. Les fichiers écrits sur `app-storage:/` à partir d'une application sont placés dans un emplacement standard :

- Sous Mac OS : le répertoire de stockage d'une application se nomme `<appData>/<appId>/Local Store/`, où `<appData>` représente le dossier des préférences de l'utilisateur. Il s'agit le plus souvent de `/Utilisateurs/<utilisateur>/Bibliothèque/Preferences`
- Sous Windows : le répertoire de stockage d'une application se nomme `<appData>/<appId>/Local Store/`, où `<appData>` représente le dossier spécial CSIDL_APPDATA de l'utilisateur. Il s'agit le plus souvent de `C:\Documents and Settings\<nom d'utilisateur>\Application Data`
- Sous Linux : `<appData>/<appID>/Local Store/`, où `<appData>` correspond à `/home/<utilisateur>/.appdata`

Si une application est conçue pour interagir avec des fichiers existants dans le système de fichiers de l'utilisateur, prenez soin de lire les « Recommandations destinées aux développeurs en matière de sécurité » à la page 1140.

Utilisation sécurisée d'un contenu non approuvé

Adobe AIR 1.0 et les versions ultérieures

Un contenu qui n'est pas affecté au sandbox de l'application peut fournir des fonctionnalités supplémentaires à votre application, mais uniquement s'il respecte les critères de sécurité du moteur d'exécution. La présente rubrique décrit le contrat de sécurité AIR avec un contenu hors application.

Security.allowDomain()

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR limitent la programmation de l'accès au contenu hors application de façon plus rigoureuse que le module d'extension du navigateur de Flash Player ne le fait pour du contenu non approuvé. Par exemple, dans le Flash Player du navigateur, lorsqu'un fichier SWF affecté au sandbox `approuvé localement` appelle la méthode `System.allowDomain()`, la programmation de l'accès est autorisée à tout SWF chargé à partir du domaine spécifié. L'approche analogue à partir du contenu de `l'application` dans les applications AIR n'est pas permise, car cela consisterait à autoriser un accès déraisonnable au fichier hors application dans le système de fichiers de l'utilisateur. Les fichiers distants ne peuvent pas accéder directement au sandbox de l'application, quels que soient les appels à la méthode `Security.allowDomain()`.

Programmation entre contenu d'application et contenu hors application

Adobe AIR 1.0 et les versions ultérieures

Les applications AIR qui programment entre contenu d'application et contenu hors application disposent de mécanismes de sécurité plus complexes. Les fichiers qui ne sont pas dans le sandbox de l'application ne sont autorisés à accéder aux propriétés et méthodes des fichiers dans le sandbox de l'application que par l'utilisation d'un pont de sandbox. Un pont de sandbox agit en tant que passerelle entre contenu de l'application et contenu hors application. Il fournit ainsi une interaction explicite entre les deux fichiers. Lorsqu'ils sont utilisés correctement, les ponts de sandbox procurent une couche supplémentaire de sécurité pour empêcher un contenu hors application d'accéder à des références d'objets qui font partie du contenu de l'application.

L'avantage que peut présenter un pont de sandbox est très bien décrit par un exemple. Supposez qu'une application de disquaire AIR souhaite fournir une interface de programmation à des publicitaires voulant créer leurs propres fichiers SWF qui leur permettront de communiquer avec l'application du disquaire. Celui-ci veut fournir aux publicitaires des méthodes pour rechercher des artistes et des CD du magasin. Mais il veut aussi isoler quelques méthodes et propriétés du fichier SWF tiers, pour des raisons de sécurité.

Cette possibilité existe avec un pont de sandbox. Par défaut, un contenu chargé de l'extérieur dans une application AIR à l'exécution n'a accès à aucune méthode ou propriété de l'application principale. Avec une implémentation de pont de sandbox personnalisé, un développeur peut fournir des services au contenu distant sans exposer ces méthodes et propriétés. Considérez le pont de sandbox comme une voie entre contenu approuvé et non approuvé qui fournit une communication entre le contenu chargé et le contenu récepteur sans présenter de références d'objet.

Pour plus d'informations sur la façon d'utiliser les ponts de sandbox en toute sécurité, voir la section « Programmation entre contenus de différents domaines » à la page 1133.

Protection contre la génération dynamique de contenu SWF à risque

Adobe AIR 1.0 et les versions ultérieures

La méthode `Loader.loadBytes()` fournit un moyen pour une application de générer un contenu SWF à partir d'un tableau d'octets. Cependant, des attaques par injection sur des données chargées à partir de sources distantes pourraient causer des dommages sérieux lors du chargement du contenu. Ceci est particulièrement valable lors du chargement des données dans le sandbox de l'application où le contenu SWF généré peut accéder à l'ensemble des interfaces de programmation d'AIR au complet.

Il existe des conditions légitimes pour utiliser la méthode `loadBytes()` sans générer de code SWF exécutable. Vous pouvez utiliser cette méthode pour générer des données d'image afin de contrôler le délai d'affichage de celle-ci, par exemple. Il y a aussi des situations légitimes qui dépendent *effectivement* de l'exécution du code, comme la création dynamique de contenu SWF pour de la lecture audio. Dans AIR, par défaut, la méthode `loadBytes()` ne vous permet *pas* de charger un contenu SWF ; il ne vous permet que de charger un contenu d'image. Dans AIR, la propriété `loaderContext` de la méthode `loadBytes()` possède une propriété `allowLoadBytesCodeExecution` que vous pouvez définir sur `true` afin d'autoriser explicitement l'application à utiliser `loadBytes()` pour charger un contenu SWF exécutable. Le code ci-dessous montre comment utiliser cette fonction :

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

Si vous appelez `loadBytes()` pour charger un contenu SWF et que la propriété `allowLoadBytesCodeExecution` de l'objet LoaderContext est définie sur `false` (valeur par défaut), l'objet Loader renvoie une exception SecurityError.

Remarque : *dans une version ultérieure d'Adobe AIR, cette interface de programmation pourrait changer. Lorsque cela se produira, il vous sera peut-être nécessaire de recompiler le contenu qui utilise la propriété* `allowLoadBytesCodeExecution` *de la classe LoaderContext.*

Recommandations destinées aux développeurs en matière de sécurité

Adobe AIR 1.0 et les versions ultérieures

Bien que les applications AIR soient créées à l'aide des technologies du Web, il est important pour les développeurs de noter qu'ils ne travaillent pas au sein du sandbox de sécurité du navigateur. Ceci signifie qu'il est possible de créer des applications qui n'endommagent pas le système local, intentionnellement ou non. AIR tente de minimiser ce risque, mais il y a toujours moyen d'introduire des vulnérabilités. La présente rubrique décrit les risques potentiels en matière de sécurité.

Risque lors de l'importation de fichiers dans un sandbox de sécurité de l'application

Adobe AIR 1.0 et les versions ultérieures

Les fichiers qui existent dans le répertoire de l'application sont affectés au sandbox de l'application et ils disposent des privilèges du moteur d'exécution au complet. Les applications qui écrivent dans le système de fichiers local devraient plutôt écrire dans `app-storage:/`. Ce répertoire est distinct des fichiers d'application puisqu'il se trouve sur l'ordinateur de l'utilisateur. De ce fait, les fichiers n'étant pas affectés au sandbox de l'application, ils présentent un risque réduit en matière de sécurité. Il est recommandé aux développeurs de tenir compte des éléments suivants :

- N'incluez un fichier dans un fichier AIR (de l'application installée) que si c'est absolument nécessaire.
- N'incluez un fichier de programmation dans un fichier AIR (de l'application installée) que si son comportement est bien compris et approuvé.
- Il ne faut pas écrire ou modifier de contenu dans le répertoire de l'application. Le moteur d'exécution empêche les applications d'écrire ou de modifier des fichiers et des répertoires à l'aide du modèle d'URL `app:/` en renvoyant une exception SecurityError.
- N'utilisez pas de données, provenant d'une source de réseau, comme paramètres pour les méthodes de l'interface de programmation AIR qui pourraient conduire à une exécution du code. Ce point est également valable pour la méthode `Loader.loadBytes()` et la fonction `eval()` de JavaScript.

Risque lors de l'utilisation d'une source externe pour déterminer des chemins

Adobe AIR 1.0 et les versions ultérieures

Une application AIR court un risque lors de l'utilisation de données ou de contenu externes. C'est pourquoi il faut prendre grand soin lors de l'utilisation de données qui proviennent du réseau ou du système de fichiers. C'est en fin de compte le développeur qui est responsable de la fiabilité des connexions réseau établies. Le chargement de données étrangères est toutefois risqué par nature et ne devrait pas être utilisé dans le cadre d'opérations critiques. Les mises en garde suivantes s'adressent aux développeurs :

- Utilisation de données provenant d'une source du réseau pour déterminer un nom de fichier
- Utilisation de données provenant d'une source du réseau pour créer une URL à laquelle l'application fait appel pour envoyer des informations de nature privée

Risques lors de l'utilisation, du stockage ou de la transmission d'informations d'identification non sécurisées

Adobe AIR 1.0 et les versions ultérieures

Le stockage des informations d'identification de l'utilisateur sur son système de fichiers local introduit par nature un grand risque. Il est recommandé aux développeurs de tenir compte des éléments suivants :

- Si les informations d'identification doivent être stockées localement, chiffrez-les lorsque vous écrivez dans le système de fichiers local. Le moteur d'exécution fournit, par la classe EncryptedLocalStore, un emplacement de stockage unique pour chaque application installée Pour plus d'informations, voir « Stockage local chiffré » à la page 737.
- Ne transmettez pas les informations d'identification de l'utilisateur non chiffrées à une source réseau, à moins que cette source soit fiable et que la transmission utilise le protocole HTTPS: ou Transport Layer Security (TLS).
- Ne spécifiez jamais un mot de passe par défaut lors de la création d'informations d'identification ; laissez l'utilisateur créer les siennes. Les utilisateurs qui ne modifient pas les éléments par défaut exposent leurs informations d'identification à un attaquant qui connaît déjà le mot de passe par défaut.

Risque d'attaque par rétrogradation de version

Adobe AIR 1.0 et les versions ultérieures

Au cours de l'installation de l'application, le moteur d'exécution s'assure qu'une version de l'application n'est pas actuellement installée. Si elle l'est déjà, le moteur d'exécution compare la chaîne de la version à la version en cours d'installation. S'il y a une différence, l'utilisateur peut choisir de mettre à niveau son installation. Le moteur d'exécution ne peut pas garantir que la version nouvellement installée soit plus récente que la précédente, seulement qu'elle est différente. Un attaquant peut distribuer une version antérieure à l'utilisateur pour contourner une faiblesse dans la sécurité. C'est pourquoi il est recommandé au développeur de vérifier la version lorsque l'application est exécutée. Il est également utile que les applications vérifient le réseau pour des mises à jour. Ainsi, même si un attaquant parvient à faire exécuter une version antérieure à un utilisateur, celle-ci se rendra compte qu'elle a besoin d'une mise à jour. De plus, la mise en place d'un modèle de contrôle des versions de l'application clair complique la tâche d'un attaquant qui vise à convaincre un utilisateur d'installer une version antérieure.

Signature du code

Adobe AIR 1.0 et les versions ultérieures

Tous les fichiers du programme d'installation AIR sont tenus de contenir du code signé. La signature du code est un processus cryptographique qui confirme que l'origine spécifiée du logiciel est exacte. Une application AIR peut être signée par le biais d'un certificat délivré par une autorité de certification externe (CA) ou d'un certificat auto-signé que vous créez vous-même. Un certificat commercial délivré par une AC bien connue est fortement recommandé. Il garantit à vos utilisateurs qu'ils installent votre application et non pas une contrefaçon. Il est cependant possible de créer des certificats auto-signés à l'aide d'`adt` à partir du kit SDK ou à l'aide de Flash, Flash Builder ou toute autre application qui utilise `adt` pour la génération de certificats. Les certificats auto-signés ne garantissent pas l'intégrité de l'application en cours d'installation et devraient être réservés aux tests d'applications réalisés avant la commercialisation de ces dernières.

Sécurité sur les périphériques Android

Adobe AIR 2.5 et les versions ultérieures

Sur Android, comme pour tout autre périphérique, AIR se conforme au modèle de sécurité natif. AIR gère par ailleurs ses propres règles de sécurité, qui ont pour objet de faciliter la programmation d'applications connectées à Internet sécurisées.

Etant donné que les applications AIR sur Android utilisent le format de package Android, l'installation relève du modèle de sécurité d'Android. Le programme d'installation des applications AIR n'est pas utilisé.

Le modèle de sécurité d'Android s'articule autour de trois éléments fondamentaux :

- Droits
- Signatures d'application
- ID utilisateur d'application

Autorisations Android

De nombreuses fonctionnalités d'Android sont protégées par le mécanisme de droits du système d'exploitation. Pour utiliser une fonctionnalité protégée, le descripteur d'application AIR doit déclarer que l'application nécessite le droit requis. Lorsqu'un utilisateur tente d'installer l'application, le système d'exploitation Android indique tous les droits requis à l'utilisateur avant que l'installation ne débute.

La plupart des applications AIR doivent spécifier les droits Android dans le descripteur d'application. Aucun droit n'est accordé par défaut. Les fonctionnalités Android protégées qui sont proposées par le moteur d'exécution d'AIR nécessitent les droits suivants :

ACCESS_COARSE_LOCATION Permet à l'application d'accéder aux données de localisation du réseau WIFI ou cellulaire via la classe Geolocation.

ACCESS_FINE_LOCATION Permet à l'application d'accéder aux données GPS via la classe Geolocation.

ACCESS_NETWORK_STATE et ACCESS_WIFI_STATE Permet à l'application d'accéder aux informations du réseau via la classe NetworkInfo.

CAMERA Permet à l'application d'accéder à la caméra.

INTERNET Permet à l'application d'effectuer des requêtes réseau. Permet également de procéder au débogage à distance.

READ_PHONE_STATE Permet au moteur d'exécution d'AIR de désactiver l'audio lorsqu'il se produit un appel entrant.

RECORD_AUDIO Permet à l'application d'accéder au microphone.

WAKE_LOCK et DISABLE_KEYGUARD Permet à l'application d'empêcher le périphérique d'entrer en mode de veille à l'aide des paramètres de la classe SystemIdleMode.

WRITE_EXTERNAL_STORAGE Permet à l'application d'écrire sur la carte mémoire externe du périphérique.

Signatures d'application

Tous les packages d'application destinés à la plate-forme Android doivent impérativement être signés. Etant donné que les applications AIR sur Android sont mises en package au format APK Android natif, elles sont signées conformément aux conventions Android, plutôt qu'aux conventions AIR. Bien qu'Android et AIR utilisent la signature de code de manière similaire, il existe des différences non négligeables :

- Sur Android, la signature vérifie que le développeur dispose de la clé privée, mais celle-ci n'est pas utilisée pour vérifier l'identité du développeur.

- Pour les applications destinées au marché Android, le certificat doit être valide pendant 25 ans au moins.
- Android ne prend pas en charge la migration de la signature d'un package vers un autre certificat. Si une mise à jour est signée par un autre certificat, les utilisateurs doivent désinstaller l'application d'origine pour pouvoir installer l'application mise à jour.
- Deux applications signées par le même certificat peuvent spécifier un ID partagé qui les autorise à accéder aux fichiers de données et au cache de l'autre application. (Ce type de partage n'est pas géré par AIR.)

ID utilisateur d'application

Android utilise un noyau Linux. A chaque application installée est affecté un ID utilisateur de type Linux, qui détermine ses droits pour des opérations telles que l'accès aux fichiers. Les fichiers stockés dans les répertoires d'application, de stockage d'application et les répertoires temporaires sont protégés par des droits de système de fichiers. Les fichiers écrits dans un emplacement de stockage externe (en d'autres termes, la carte SD) peuvent être lus, modifiés et supprimés par d'autres applications ou par l'utilisateur lorsque la carte SD est montée en tant que périphérique de stockage de masse sur un ordinateur.

Les cookies associés aux requêtes Internet ne sont pas partagés entre applications AIR.

Confidentialité de l'image d'arrière-plan

Lorsqu'un utilisateur fait passer une application en arrière-plan, certaines versions d'Android effectuent une capture d'écran qu'elles utilisent comme miniature dans la liste des applications récentes. Cette capture d'écran est stockée dans la mémoire du périphérique et tout attaquant qui contrôle physiquement ce dernier peut y accéder.

Si l'application affiche des informations confidentielles, celles-ci ne doivent pas figurer sur la capture d'écran d'arrière-plan. L'événement `deactivate` distribué par l'objet NativeApplication indique qu'une application est sur le point de passer en arrière-plan. Effacez ou masquez toute information confidentielle à l'aide de cet événement.

Voir aussi

Android : Security and Permissions

Données chiffrées sur Android

Les applications AIR sur Android disposent des options de chiffrement proposées par la base de données SQL intégrée pour enregistrer les données chiffrées. Pour une sécurité optimale, basez la clé de chiffrement sur un mot de passe que l'utilisateur devra saisir lors de chaque exécution de l'application. Une clé de chiffrement (ou un mot de passe) enregistrée localement est difficile, voir impossible à cacher à un attaquant qui a accès aux fichiers d'application. Si l'attaquant est en mesure de récupérer la clé, le chiffrement des données ne permet aucune autre protection que la sécurité du système de fichiers basée sur l'ID utilisateur fournie par le système Android.

Il est possible d'utiliser la classe EncryptedLocalStore pour enregistrer les données, mais celles-ci ne sont pas chiffrées sur les périphériques Android. Le modèle de sécurité d'Android repose sur l'ID utilisateur de l'application pour protéger les données des autres applications. Les applications qui utilisent un ID utilisateur partagé et qui sont signées avec le même certificat de signature de code font appel au même magasin local chiffré.

Important : *sur un téléphone associé à une racine, toute application s'exécutant avec des privilèges racines peut accéder aux fichiers d'autres applications. Les données enregistrées à l'aide du magasin local chiffré ne sont donc pas sécurisées sur un périphérique associé à une racine.*

Sécurité sur les périphériques iOS

Sous iOS, AIR adopte le modèle de sécurité natif. AIR gère par ailleurs ses propres règles de sécurité, qui ont pour objet de faciliter la programmation d'applications connectées à Internet sécurisées.

Etant donné que les applications AIR sous iOS utilisent le format de package iOS, l'installation répond au modèle de sécurité d'iOS. Le programme d'installation des applications AIR n'est pas utilisé. Par ailleurs, aucun moteur d'exécution AIR distinct n'est utilisé sur les périphériques iOS. Chaque application AIR contient tout le code nécessaire pour fonctionner.

Signatures d'application

Tous les packages d'application créés pour la plate-forme iOS doivent être signés. Etant donné que les applications AIR sous iOS sont mises en package au format IPA natif d'iOS, elles sont signées conformément aux exigences d'iOS et non aux exigences d'AIR. Bien qu'iOS et AIR utilisent de façon similaire la signature de code, il existe des différences significatives :

- Sous iOS, le certificat utilisé pour signer une application doit être émis par Apple ; il est impossible d'utiliser des certificats émanant d'autres autorités de certification.
- Sous iOS, les certificats de distribution émis par Apple sont normalement valides pendant un an.

Confidentialité de l'image d'arrière-plan

Lorsqu'une application passe en arrière-plan sous iOS, le système d'exploitation effectue une capture d'écran permettant d'animer la transition. Cette capture d'écran est stockée dans la mémoire du périphérique et tout attaquant qui contrôle physiquement ce dernier peut y accéder.

Si l'application affiche des informations confidentielles, celles-ci ne doivent pas figurer sur la capture d'écran d'arrière-plan. L'événement `deactivate` distribué par l'objet NativeApplication indique qu'une application est sur le point de passer en arrière-plan. Effacez ou masquez toute information confidentielle à l'aide de cet événement.

Chapitre 65 : Utilisation des exemples de code ActionScript

Le meilleur moyen pour comprendre le fonctionnement des classes et des méthodes est d'exécuter un exemple de code ActionScript 3.0. Vous pouvez recourir aux exemples de diverses façons, en fonction des périphériques que vous utilisez ou ciblez.

Ordinateurs exécutant Flash Professional ou Flash Builder Voir « Exécution d'exemples ActionScript 3.0 dans Flash Professional » à la page 1147 ou « Exécution d'exemples ActionScript 3.0 dans Flash Builder » à la page 1148 pour plus d'informations sur le mode d'utilisation de ces environnements de développement en vue d'exécuter les exemples ActionScript 3.0. Utilisez les instructions trace et d'autres outils de débogage pour mieux comprendre le fonctionnement d'un exemple de code.

Périphériques mobiles Vous pouvez exécuter les exemples de code ActionScript 3.0 sur des périphériques mobiles qui prennent en charge Flash Player 10.1 et les versions ultérieures. Voir « Exécution d'exemples ActionScript 3.0 sur un périphérique mobile » à la page 1150. Vous pouvez par ailleurs exécuter ces exemples sur votre ordinateur à l'aide de Flash Professional ou de Flash Builder.

Périphériques TV Bien que vous ne puissiez pas exécuter ces exemples sur des périphériques TV, vous pouvez en tirer parti en les exécutant sur votre ordinateur. Voir Flash Platform for TV sur le site Web Adobe Developer Connection pour plus d'informations sur le développement d'applications pour périphériques TV.

Types d'exemples

Les types d'exemples de code ActionScript 3.0 sont les suivants :

- Exemples de fragments de code (dans l'ensemble de la documentation d'ActionScript 3.0)
- Exemples basés sur les classes (principalement dans le Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash)
- Exemples pratiques contenant plusieurs fichiers sources (télécharger les fichiers sources ZIP depuis www.adobe.com/go/learn_programmingAS3samples_flash_fr)

Exemples de type fragment de code

Un exemple de type fragment de code se présente comme suit :

```
var x:int = 5;
trace(x); // 5
```

Les fragments de code ne contiennent que le code nécessaire à l'illustration d'un concept unique. Ils ne comportent généralement pas d'instruction de package ou de classe.

Exemples basés sur les classes

De nombreux exemples illustrent le code source associé à une classe ActionScript entière. Un exemple basé sur une classe se présente comme suit :

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

Le code d'un exemple basé sur une classe comprend une instruction de package, une déclaration de classe et une fonction de constructeur.

Exemples pratiques contenant plusieurs fichiers source

La plupart des rubriques du Guide du développeur d'ActionScript 3.0 se concluent par des exemples pratiques qui illustrent l'utilisation de certaines fonctionnalités ActionScript dans un contexte pratique et réel. Ces exemples comportent généralement plusieurs fichiers, à savoir :

- Un ou plusieurs fichiers ActionScript source
- Un fichier FLA à utiliser avec Flash Professional
- Un ou plusieurs fichiers MXML à utiliser avec Flash Builder
- Des fichiers de données, d'image et de son ou autres actifs utilisés par l'exemple d'application (facultatifs)

Ces exemples sont le plus souvent livrés sous forme de fichiers d'archives ZIP.

Liste des exemples du guide du développeur dans le fichier ZIP

Le fichier ZIP pour Flash Professional CS5 et Flex 4 (à télécharger depuis www.adobe.com/go/learn_programmingAS3samples_flash_fr) contient les exemples suivants :

- AlarmClock (« Exemple de gestion des événements : Alarm Clock » à la page 145)
- AlgorithmicVisualGenerator (« Exemple d'utilisation de l'API de dessin : générateur algorithmique d'effets visuels » à la page 241)
- ASCIIArt (« Exemple de chaîne : ASCII Art » à la page 20)
- CapabilitiesExplorer (« Exemple d'utilisation de Capabilities : détection des capacités du système » à la page 906)
- CustomErrors (« Exemple de gestion des erreurs : application CustomErrors » à la page 72)
- DisplayObjectTransformer (« Exemple de géométrie : application d'une transformation de matrice à un objet d'affichage » à la page 226)
- FilterWorkbench (« Exemple de filtrage des objets d'affichage : Filter Workbench » à la page 302)
- GlobalStockTicker (« Exemple : Internationalisation d'une application de suivi des stocks » à la page 991)
- IntrovertIM_HTML (« Exemple d'API externe : communications entre ActionScript et JavaScript dans un navigateur Web » à la page 882)
- NewsLayout (« Exemple TextField : mise en forme du texte dans le style « article de journal » » à la page 401)
- PlayList (« Exemple de tableau : PlayList » à la page 49)
- PodcastPlayer (« Exemple d'objet Sound : Podcast Player » à la page 482)
- ProjectionDragger (« Exemple : Projection de perspective » à la page 369)
- ReorderByZ (« Réorganisation de l'affichage à l'aide d'objets Matrix3D » à la page 374)
- RSSViewer (« Utilisation de XML dans un exemple ActionScript : chargement de données RSS depuis Internet » à la page 118)

- RuntimeAssetsExplorer (« Exemple de clip : RuntimeAssetsExplorer » à la page 342)
- SimpleClock (« Exemple de date et heure : horloge analogique simple » à la page 6)
- SpinningMoon (« Exemple d'objet Bitmap : lune en rotation animée » à la page 263)
- SpriteArranger (« Exemple d'objet d'affichage : SpriteArranger » à la page 211)
- TelnetSocket (« Exemple de socket TCP : création d'un client Telnet » à la page 830)
- VideoJukebox (« Exemple vidéo : Video Jukebox » à la page 523)
- WikiEditor (« Exemple d'expression régulière : analyseur Wiki » à la page 94)
- WordSearch (« Exemple d'entrée de souris : WordSearch » à la page 596)

De nombreux articles de démarrage rapide figurant dans le Centre des développeurs Flash et Flex comportent également des exemples pratiques.

Exécution d'exemples ActionScript 3.0 dans Flash Professional

Appliquez l'une des procédures suivantes (selon le type d'exemple) pour exécuter un exemple à l'aide de Flash Professional.

Exécution d'un exemple de fragment de code dans Flash Professional

Pour exécuter un exemple de fragment de code dans Flash Professional :

1. Choisissez Fichier > Nouveau.
2. Dans la boîte de dialogue Nouveau document, sélectionnez Document Flash et cliquez sur OK.

   Une nouvelle fenêtre Flash apparaît.
3. Cliquez sur la première image du premier calque dans le panneau Scénario.
4. Dans le panneau Actions, saisissez ou collez l'exemple de fragment de code.
5. Choisissez Fichier > Enregistrer. Attribuez un nom au fichier et cliquez sur OK.
6. Pour tester l'exemple, sélectionnez Contrôle > Tester l'animation.

Exécution d'un exemple basé sur une classe dans Flash Professional

Pour exécuter un exemple basé sur une classe dans Flash Professional :

1. Sélectionnez Fichier > Nouveau.
2. Dans la boîte de dialogue Nouveau document, sélectionnez un fichier ActionScript, puis cliquez sur OK. Une nouvelle fenêtre d'édition apparaît.
3. Copiez l'exemple de code basé sur une classe et collez-le dans la fenêtre d'édition.

   S'il s'agit de la classe de document principale du programme, la classe doit étendre la classe MovieClip :

```
import flash.display.MovieClip;
public class Example1 extends MovieClip{
//...
}
```

Assurez-vous également que toutes les classes auxquelles fait référence l'exemple sont déclarées à l'aide d'instructions `import`.

4 Sélectionnez Fichier > Enregistrer. Attribuez au fichier le nom de la classe stipulée dans l'exemple (tel ExempleMenuContextuel.as).

Remarque : *certains exemples basés sur les classes, tels que l'exemple de la classe flashx.textLayout.container.ContainerController, comprennent plusieurs niveaux dans la déclaration du package (`package flashx.textLayout.container.examples {`). Pour ces exemples, enregistrez le fichier dans un sous-dossier qui doit correspondre à la déclaration du package (flashx/textLayout/container/examples) ou supprimez le nom du package (afin qu'ActionScript démarre avec le `package {` uniquement) pour tester le fichier depuis n'importe quel emplacement.*

5 Sélectionnez Fichier > Nouveau.

6 Dans la boîte de dialogue Nouveau document, sélectionnez Document Flash (ActionScript 3.0) et cliquez sur OK. Une nouvelle fenêtre Flash apparaît.

7 Dans le champ Classe du document du panneau Propriétés, entrez le nom de la classe de l'exemple, qui doit correspondre au nom du fichier source ActionScript que vous venez d'enregistrer (ExempleMenuContextuel, par exemple).

8 Sélectionnez Fichier > Enregistrer. Attribuez au fichier FLA le nom de la classe stipulée dans l'exemple (tel ExempleMenuContextuel.fla).

9 Pour tester l'exemple, sélectionnez Contrôle > Tester l'animation.

Exécution d'un exemple pratique dans Flash Professional

Les exemples pratiques sont le plus souvent livrés sous forme de fichiers d'archives ZIP. Pour exécuter un exemple pratique dans Flash Professional :

1 Désarchivez le fichier ZIP dans le dossier de votre choix.

2 Dans Flash Professional, sélectionnez Fichier > Ouvrir.

3 Accédez au dossier où vous avez désarchivé le fichier ZIP. Sélectionnez le fichier FLA dans ce dossier, puis cliquez sur Ouvrir.

4 Pour tester l'exemple, sélectionnez Contrôle > Tester l'animation.

Exécution d'exemples ActionScript 3.0 dans Flash Builder

Appliquez l'une des procédures suivantes (selon le type d'exemple) pour exécuter un exemple à l'aide de Flash Builder.

Exécution d'un exemple de fragment de code dans Flash Builder

Pour exécuter un exemple de fragment de code dans Flash Builder :

1 Créez un projet Flex (en sélectionnant File > New > Flex Project) ou créez une application MXML dans un projet Flex existant (en sélectionnant File > New > MXML Application). Attribuez au projet ou à l'application un nom descriptif tel que ExempleMenuContextuel.

2 Dans le fichier MXML généré, ajoutez une balise `<mx:Script>`.

3 Collez le contenu de l'exemple de fragment de code entre les balises `<mx:Script>` et `</mx:Script>`. Enregistrez le fichier MXML.

4 Pour exécuter l'exemple, sélectionnez l'option de menu Run > Run associée au principal fichier MXML (Run > Run ExempleMenuContextuel, par exemple).

Exécution d'un exemple basé sur une classe dans Flash Builder

Pour exécuter un exemple basé sur une classe dans Flash Builder :

1 Sélectionnez File > New > ActionScript Project.

2 Entrez le nom de la classe principale (ExempleMenuContextuel, par exemple) dans le champ Project Name. Utilisez les valeurs par défaut des autres champs ou modifiez-les en fonction de l'environnement en vigueur. Cliquez sur Finish pour créer le projet et le principal fichier ActionScript.

3 Effacez tout contenu généré du fichier ActionScript. Collez l'exemple de code, y compris les instructions de package et d'importation, dans le fichier ActionScript et enregistrez ce dernier.

Remarque : *certains exemples basés sur les classes, tels que l'exemple de la classe flashx.textLayout.container.ContainerController, comprennent plusieurs niveaux dans la déclaration du package (*`package flashx.textLayout.container.examples {`*). Pour ces exemples, enregistrez le fichier dans un sous-dossier qui doit correspondre à la déclaration du package (flashx/textLayout/container/examples) ou supprimez le nom du package (afin qu'ActionScript démarre avec le* `package {` *uniquement) pour tester le fichier depuis n'importe quel emplacement.*

4 Pour exécuter l'exemple, sélectionnez l'option de menu Run > Run associée au nom de la classe ActionScript principale (Run > Run ExempleMenuContextuel, par exemple).

Exécution d'un exemple pratique dans Flash Builder

Les exemples pratiques sont le plus souvent livrés sous forme de fichiers d'archives ZIP. Pour exécuter un exemple pratique dans Flash Builder :

1 Désarchivez le fichier ZIP dans le dossier de votre choix. Attribuez au dossier un nom descriptif tel qu'ExempleMenuContextuel.

2 Dans Flash Builder, sélectionnez File > New Flex Project. Dans la section Project Location, cliquez sur Browse et sélectionnez le dossier contenant les fichiers d'exemple. Dans le champ Project Name, entrez le nom du dossier (ExempleMenuContextuel, par exemple). Utilisez les valeurs par défaut des autres champs ou modifiez-les en fonction de l'environnement en vigueur. Cliquez sur Next pour poursuivre la procédure.

3 Dans le panneau Output, cliquez sur Next pour accepter la valeur par défaut.

4 Dans le panneau Source Paths, cliquez sur le bouton Browse en regard du champ Main Application File. Sélectionnez le principal fichier MXML dans le dossier d'exemples. Cliquez sur Finish pour créer les fichiers de projet.

5 Pour exécuter l'exemple, sélectionnez l'option de menu Run > Run associée au principal fichier MXML (Run > Run ExempleMenuContextuel, par exemple).

Exécution d'exemples ActionScript 3.0 sur un périphérique mobile

Vous pouvez exécuter des exemples de code ActionScript 3.0 sur un périphérique mobile qui prend en charge Flash Player 10.1. Ceci dit, cette opération a généralement pour objet d'apprendre à utiliser des classes et méthodes déterminées. Dans ce cas de figure, exécutez l'exemple sur un périphérique non mobile tel qu'un ordinateur de bureau. L'ordinateur de bureau permet de faire appel à des instructions de suivi et autres outils de débogage Flash Professional ou Flash Builder pour se familiariser avec un exemple de code.

Pour exécuter l'exemple sur un périphérique mobile, copiez les fichiers correspondant sur ce dernier ou sur un serveur Web. Pour copier les fichiers sur le périphérique et exécuter l'exemple dans le navigateur, procédez comme suit :

1. Créez le fichier SWF en vous référant aux instructions stipulées dans « Exécution d'exemples ActionScript 3.0 dans Flash Professional » à la page 1147 ou « Exécution d'exemples ActionScript 3.0 dans Flash Builder » à la page 1148. Dans Flash Professional, vous créez le fichier SWF par le biais de Contrôle > Tester l'animation. Dans Flash Builder, vous créez le fichier SWF lors de l'exécution, du débogage ou de la création du projet Flash Builder.
2. Copiez le fichier SWF dans un répertoire du périphérique mobile. Utilisez à cet effet le logiciel fourni avec le périphérique mobile.
3. Dans la barre d'adresse du navigateur sur le périphérique mobile, entrez la chaîne « file:// URL » associée au fichier SWF. Entrez par exemple `file://applications/MonExemple.swf`.

Pour copier les fichiers sur un serveur Web et exécuter l'exemple dans le navigateur du périphérique, procédez comme suit :

1. Créez un fichier SWF et un fichier HTML. Commencez par suivre les instructions figurant dans « Exécution d'exemples ActionScript 3.0 dans Flash Professional » à la page 1147 ou « Exécution d'exemples ActionScript 3.0 dans Flash Builder » à la page 1148. Dans Flash Professional, la sélection de Contrôle > Tester l'animation crée le fichier SWF uniquement. Pour créer les deux fichiers, commencez par sélectionner à la fois Flash et HTML sur l'onglet Formats de la boîte de dialogue Paramètres de publication. Sélectionnez ensuite Fichier > Publier pour créer à la fois le fichier HTML et le fichier SWF. Dans Flash Builder, vous créez le fichier SWF et le fichier HTML lors de l'exécution, du débogage ou de la création du projet Flash Builder.
2. Copiez les fichiers SWF et HTML dans un répertoire hébergé sur le serveur Web.
3. Dans la barre d'adresse du navigateur du périphérique mobile, entrez l'adresse HTTP du fichier HTML. Entrez par exemple `http://www.MonServeurWeb/exemples/MonExemple.html`.

Avant d'exécuter un exemple sur un périphérique mobile, tenez compte des considérations suivantes.

Taille de la scène

La taille de scène activée lorsque vous exécutez un exemple sur un périphérique mobile est considérablement inférieure à la taille en vigueur sur un périphérique non mobile. De nombreux exemples ne nécessitent pas une taille de scène spécifique. Lorsque vous créez le fichier SWF, stipulez une taille de scène adaptée au périphérique (176 x 208 pixels, par exemple).

Les exemples pratiques figurant dans le Guide du développeur d'Adobe ActionScript 3.0 ont pour objet d'illustrer divers concepts et classes ActionScript 3.0. Tant au niveau présentation qu'au niveau fonctionnement, leur interface utilisateur est adaptée à un ordinateur de bureau ou un ordinateur portable. Bien que les exemples fonctionnent sur un périphérique mobile, la taille de la scène et la conception de l'interface utilisateur ne conviennent pas à un écran de taille réduite. Adobe recommande d'exécuter les exemples pratiques sur un ordinateur pour se familiariser avec le code ActionScript, puis d'utiliser des fragments de code appropriés dans les applications mobiles.

Substitution de champs de texte aux instructions de suivi

Lorsque vous exécutez un exemple sur un périphérique mobile, il est impossible de visualiser le résultat des instructions de suivi correspondantes. Pour visualiser le résultat, créez une occurrence de la classe TextField. Ajoutez ensuite le texte issu des instructions de suivi à la fin de la propriété `text` du champ de texte.

La fonction suivante permet de définir un champ de texte à des fins de suivi :

```
function createTracingTextField(x:Number, y:Number,
                                width:Number, height:Number):TextField {

    var tracingTF:TextField = new TextField();
    tracingTF.x = x;
    tracingTF.y = y;
    tracingTF.width = width;
    tracingTF.height = height;

    // A border lets you more easily see the area the text field covers.
    tracingTF.border = true;
    // Left justifying means that the right side of the text field is automatically
    // resized if a line of text is wider than the width of the text field.
    // The bottom is also automatically resized if the number of lines of text
    // exceed the length of the text field.
    tracingTF.autoSize = TextFieldAutoSize.LEFT;

    // Use a text size that works well on the device.
    var myFormat:TextFormat = new TextFormat();
    myFormat.size = 18;
    tracingTF.defaultTextFormat = myFormat;

    addChild(tracingTF);
    return tracingTF;
}
```

Ajoutez par exemple cette fonction à la classe du document en tant que fonction privée. Pour les autres méthodes de la classe du document, effectuez le suivi des données par le biais d'un code similaire à celui-ci :

```
var traceField:TextField = createTracingTextField(10, 10, 150, 150);
// Use the newline character "\n" to force the text to the next line.
traceField.appendText("data to trace\n");
traceField.appendText("more data to trace\n");
// Use the following line to clear the text field.
traceField.appendText("");
```

La méthode `appendText()` n'accepte qu'une seule valeur à titre de paramètre. Cette valeur est une chaîne (occurrence de String ou littéral de chaîne). Pour imprimer la valeur d'une variable qui n'est pas une chaîne, commencez par convertir la valeur en chaîne. A cet effet, la technique la plus simple consiste à appeler la méthode `toString()` de l'objet :

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

Taille du texte

De nombreux exemples font appel aux champs de texte pour illustrer un concept. Régler la taille du contenu d'un champ de texte optimise parfois sa lisibilité sur un périphérique mobile. Ainsi, si un exemple fait appel à une occurrence de champ de texte appelée `myTextField`, le code suivant permet de modifier la taille du contenu :

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

Capture des entrées utilisateur

Le système d'exploitation et le navigateur du périphérique mobile capturent certains événements de type entrée utilisateur que ne reçoit pas le contenu SWF. Bien que le comportement varie selon le système d'exploitation et le navigateur, l'exécution d'exemples sur un périphérique mobile risque d'entraîner des résultats inattendus. Pour plus d'informations, voir « Priorité des événements KeyboardEvent » à la page 580.

L'interface utilisateur de nombreux exemples est par ailleurs adaptée à un ordinateur de bureau ou un ordinateur portable. Ainsi, la plupart des exemples pratiques figurant dans le Guide du développeur d'ActionScript 3.0 sont conçus pour être visualisés sur un ordinateur de bureau. De ce fait, il est parfois impossible d'afficher la scène dans son intégralité sur l'écran d'un périphérique mobile. Les fonctionnalités de défilement du contenu du navigateur varient selon ce dernier. En outre, les exemples ne sont pas conçus pour capturer et traiter les événements de défilement horizontal ou vertical. L'interface utilisateur de certains exemples n'est de ce fait pas adaptée à un écran de petite taille. Adobe recommande d'exécuter les exemples sur un ordinateur pour se familiariser avec le code ActionScript, puis d'utiliser des fragments de code appropriés dans les applications mobiles.

Pour plus d'informations, voir « Défilement horizontal ou vertical des objets d'affichage » à la page 185.

Gestion du focus

Certains exemples nécessitent l'attribution du focus à un champ. Vous pouvez ainsi entrer du texte ou sélectionner un bouton. Pour ce faire, faites appel au périphérique de pointage du périphérique mobile, tel un stylet ou le doigt. Vous disposez également des touches de navigation du périphérique mobile pour attribuer le focus à un champ. Pour sélectionner un bouton doté du focus, utilisez la touche de sélection du périphérique mobile à l'instar de la touche Entrée sur un ordinateur. Sur certains périphériques, il suffit de taper deux fois sur un bouton pour le sélectionner.

Pour plus d'informations sur le focus, voir « Gestion de la cible d'action » à la page 575.

Gestion des événements de souris

De nombreux exemples écoutent les événements de souris. Sur un ordinateur, ces événements de souris se produisent, par exemple, lorsqu'un utilisateur survole un objet d'affichage ou clique sur un objet d'affichage à l'aide de la souris. Sur un périphérique mobile, les événements issus de périphériques de pointage tels qu'un stylet ou un doigt portent le nom d'événements tactiles. Flash Player 10.1 mappe les événements tactiles sur les événements de souris. Ce mappage assure le fonctionnement du contenu SWF développé avant Flash Player 10.1. Par conséquent, les exemples fonctionnent lorsque l'utilisateur sélectionne ou fait glisser un objet d'affichage à l'aide d'un périphérique de pointage.

Performances

La puissance de traitement d'un périphérique mobile est inférieure à celle d'un ordinateur de bureau. Certains exemples qui sollicitent intensivement l'unité centrale risquent de s'exécuter lentement sur un périphérique mobile. Ainsi, l'exemple figurant dans « Exemple d'utilisation de l'API de dessin : générateur algorithmique d'effets visuels » à la page 241 effectue un grand nombre d'opérations de calcul et de dessin à chaque image. L'exécution de cet exemple sur un ordinateur illustre diverses API de dessin. L'exemple n'est toutefois pas adapté à certains périphériques mobiles, en raison de leurs performances limitées.

Pour plus d'informations sur les performances d'une application sur un périphérique mobile, voir *Optimisation des performances pour la plate-forme Adobe Flash*.

Recommandations d'utilisation

Les exemples ne tiennent pas compte des recommandations d'utilisation en matière de développement d'applications pour un périphérique mobile. Considérez les limites de mémoire et de puissance de traitement des périphériques mobiles. De même, l'interface utilisateur destinée à un écran de taille réduite possède des caractéristiques différentes de celles d'une interface pour écran d'ordinateur de bureau. Pour plus d'informations sur le développement d'applications pour périphérique mobile, voir *Optimisation des performances pour la plate-forme Adobe Flash*.

Chapitre 66 : Prise en charge de SQL dans les bases de données locales

Adobe AIR comprend un moteur de base de données SQL avec prise en charge des bases de données SQL locales et de nombreuses fonctionnalités SQL standard, à l'aide du système de base de données SQLite open source. Le moteur d'exécution ne spécifie ni comment, ni où les données de base de données sont stockées dans le système de fichiers. Chaque base de données est stockée dans son intégralité dans un seul fichier. Un développeur peut spécifier l'emplacement de stockage du fichier de base de données dans le système de fichiers et une application AIR unique peut accéder à une ou plusieurs bases de données (en d'autres termes, à des fichiers de base de données distincts). Ce document présente la syntaxe SQL et la prise en charge des types de données pour les bases de données SQL locales Adobe AIR. Ce document n'a pas pour objectif de servir de référence SQL complète, mais de décrire les détails spécifiques du langage SQL pris en charge par Adobe AIR. Le moteur d'exécution prend en charge la plupart des éléments du langage SQL standard SQL-92. Au vu du nombre de références, sites Web, manuels et supports de formation consacrés à l'apprentissage de SQL, ce document n'a pas pour objectif d'être une référence ou un didacticiel SQL complet. Il se concentre plutôt sur la syntaxe SQL gérée par AIR, ainsi que sur les différences entre SQL-92 et le langage SQL pris en charge.

Conventions utilisées dans les définitions d'instruction SQL

Les conventions suivantes sont utilisées dans les définitions des instructions du présent document :

- Casse du texte
  - MAJUSCULES : les mots-clés SQL littéraux sont écrits en majuscules.
  - minuscules : les noms de clause et les termes d'espaces réservés sont écrits en minuscules.
- Caractères de définition
  - ::= indique une définition de clause ou d'instruction.
- Caractères de regroupement et de remplacement
  - | Le caractère pipe est utilisé entre les diverses options gérées et peut être lu comme « ou ».
  - [] Les éléments placés entre crochets sont des éléments facultatifs ; les crochets peuvent contenir un seul élément ou un ensemble d'éléments gérés.
  - () Les parenthèses qui entourent un ensemble d'alternatives (ensemble d'éléments séparés par des caractères pipe) désignent un groupe d'éléments obligatoires, c'est-à-dire un ensemble d'éléments correspondant aux valeurs gérées pour un seul élément requis.
- Quantificateurs
  - + Un caractère plus suivant un élément entre parenthèses indique que l'élément précédent peut survenir une ou plusieurs fois.
  - * Un caractère astérisque suivant un élément placé entre crochets indique que l'élément précédent (entre crochets) peut survenir 0 ou plusieurs fois.
- Caractères littéraux
  - * Un caractère astérisque utilisé dans un nom de colonne ou entre des parenthèses après un nom de fonction désigne un caractère astérisque littéral et non le quantificateur « 0 ou plus ».
  - . Un caractère point représente un point littéral.

- , Un caractère virgule représente une virgule littérale.
- () Une paire de parenthèses renfermant une seule clause ou un seul élément indique que les parenthèses sont des caractères littéraux obligatoires.
- Sauf mention contraire, les autres caractères représentent ces caractères littéraux.

Syntaxe SQL prise en charge

Les listes de syntaxe SQL suivantes sont prises en charge par le moteur de base de données SQL d'Adobe AIR. Elles sont divisées en explications de divers types d'instructions et de clauses, d'expressions, de fonctions intégrées et d'opérateurs. Les sujets suivants sont passés en revue :

- Syntaxe SQL générale
- Instructions de manipulation des données (SELECT, INSERT, UPDATE et DELETE)
- Instructions de définition des données (CREATE, ALTER et instructions DROP associées aux tables, aux index, aux vues et aux déclencheurs)
- Instructions et clauses spéciales
- Fonctions intégrées (fonctions d'agrégation, fonctions scalaires et fonctions de formatage de la date et de l'heure)
- Opérateurs
- Paramètres
- Fonctions SQL non prises en charge
- Autres fonctions SQL

Syntaxe SQL générale

Outre la syntaxe spécifique des diverses instructions et expressions, les règles générales de la syntaxe SQL sont les suivantes :

Respect de la casse Les instructions SQL, y compris les noms d'objet, ne sont pas sensibles à la casse. Néanmoins, les instructions SQL sont fréquemment écrites avec des mots-clés SQL en majuscules et ce document utilise cette convention. Bien que la syntaxe SQL ne soit pas sensible à la casse, les valeurs de texte littéral incluses dans une requête SQL le sont, et les opérations de comparaison et de tri respectent parfois la casse, selon les spécifications de la séquence de classement définie pour une colonne ou une opération. Pour plus d'informations, voir COLLATE.

Espace blanc Un caractère espace blanc (espace, tabulation, nouvelle ligne, etc.) doit séparer chaque mot dans une instruction SQL. Toutefois, cet espace blanc est facultatif entre les mots et les symboles Le type et la quantité de caractères d'espace blanc dans une instruction SQL ne sont pas significatifs. Vous pouvez utiliser un espace blanc, par exemple une mise en retrait et des sauts de ligne, pour formater des instructions SQL et améliorer leur lisibilité, sans que le sens de l'instruction n'en soit modifié.

Instructions de manipulation des données

Les instructions de manipulation des données sont les instructions SQL les plus courantes. Ces instructions sont utilisées pour récupérer, ajouter, modifier et supprimer des données dans des tables de bases de données. Les instructions de manipulation de données suivantes sont prises en charge : SELECT, INSERT, UPDATE et DELETE.

SELECT

L'instruction SELECT permet d'interroger la base de données. Le résultat d'une instruction SELECT est zéro ou plusieurs lignes de données, chaque ligne contenant un nombre fixe de colonnes. Le nombre de colonnes du résultat est indiqué par le nom de la colonne de résultat ou la liste d'expressions entre SELECT et les mots-clés facultatifs FROM.

```
sql-statement   ::=  SELECT [ALL | DISTINCT] result
                     [FROM table-list]
                     [WHERE expr]
                     [GROUP BY expr-list]
                     [HAVING expr]
                     [compound-op select-statement]*
                     [ORDER BY sort-expr-list]
                     [LIMIT integer [( OFFSET | , ) integer]]
result          ::=  result-column [, result-column]*
result-column   ::=  * | table-name . * | expr [[AS] string]
table-list      ::=  table [ join-op table join-args ]*
table           ::=  table-name [AS alias] |
                     ( select ) [AS alias]
join-op         ::=  , | [NATURAL] [LEFT | RIGHT | FULL] [OUTER | INNER | CROSS] JOIN
join-args       ::=  [ON expr] [USING ( id-list )]
compound-op     ::=  UNION | UNION ALL | INTERSECT | EXCEPT
sort-expr-list  ::=  expr [sort-order] [, expr [sort-order]]*
sort-order      ::=  [COLLATE collation-name] [ASC | DESC]
collation-name  ::=  BINARY | NOCASE
```

Toute expression arbitraire peut servir de résultat. Si une expression de résultat est *, toutes les colonnes de toutes les tables sont substituées à cette expression. Si l'expression correspond à un nom de table suivi de .*, le résultat est l'ensemble des colonnes de cette table.

Le mot-clé DISTINCT entraîne le renvoi d'un sous-ensemble de lignes de résultat, dans lequel chaque ligne résultante est différente. Toutes les valeurs NULL sont considérées comme identiques. Le comportement par défaut consiste à renvoyer toutes les lignes de résultat, ce que le mot-clé ALL peut rendre explicite.

La requête est exécutée sur une ou plusieurs tables spécifiées après le mot-clé FROM. Si plusieurs noms de table sont séparés par des virgules, la requête utilise alors la jointure croisée des différentes tables. La syntaxe JOIN permet également de spécifier le mode de jointure des tables. L'unique type de jointure externe pris en charge est LEFT OUTER JOIN. L'expression de clause ON dans les éléments join-arg doit donner une valeur booléenne. Une sous-requête placée entre parenthèses peut être utilisée comme table dans la clause FROM. La clause FROM peut être omise dans sa totalité, auquel cas le résultat est une ligne unique composée des valeurs de la liste d'expressions.

La clause WHERE permet de restreindre le nombre de lignes récupérées par la requête. Les expressions de clause WHERE doivent donner une valeur booléenne. Un filtrage des clauses WHERE étant effectué avant tout regroupement, les expressions de la clause WHERE risquent de ne pas inclure de fonction d'agrégation.

La clause GROUP BY combine une ou plusieurs lignes du résultat en une ligne de résultat unique. Une clause GROUP BY se révèle particulièrement utile lorsque le résultat contient des fonctions d'agrégation. Les expressions de la clause GROUP BY ne doivent pas nécessairement apparaître dans la liste d'expressions SELECT.

La clause HAVING est similaire à la clause WHERE, car elle limite le nombre de lignes renvoyées par l'instruction. La clause HAVING est toutefois appliquée après l'exécution d'un regroupement spécifié par une clause GROUP BY. Par conséquent, l'expression HAVING peut faire référence aux valeurs qui incluent des fonctions d'agrégation. Une expression de clause HAVING ne doit pas obligatoirement apparaître dans la liste SELECT. Telle une expression WHERE, une expression HAVING doit donner une valeur booléenne.

La clause ORDER BY entraîne le tri des lignes de résultat. L'argument sort-expr-list de la clause ORDER BY est une liste d'expressions servant de clé du tri. Il n'est pas nécessaire que les expressions fassent partie du résultat d'une instruction SELECT simple, mais dans une instruction SELECT composée (instruction SELECT utilisant l'un des opérateurs compound-op), chaque expression de tri doit correspondre exactement à l'une des colonnes de résultats. Chaque expression de tri peut éventuellement être suivie d'une clause sort-order composée du mot-clé COLLATE et du nom d'une fonction de classement utilisée pour organiser le texte et/ou du mot-clé ASC ou DESC pour spécifier l'ordre de tri (croissant ou décroissant). Vous pouvez omettre la clause sort-order, auquel cas l'ordre par défaut (croissant) est utilisé. Pour consulter une définition de la clause COLLATE et des fonctions de classement, voir COLLATE.

La clause LIMIT définit le nombre maximal de lignes de résultat renvoyées. Une clause LIMIT négative indique qu'il n'existe pas de limite supérieure. Le mot-clé OFFSET facultatif qui suit la clause LIMIT spécifie le nombre de lignes à ignorer au début du jeu de résultats. Dans une requête SELECT composée, la clause LIMIT peut n'apparaître qu'après l'instruction SELECT finale, et la limite s'applique à l'ensemble de la requête. Notez que si le mot-clé OFFSET est utilisé dans la clause LIMIT, la limite correspond au premier nombre entier et le décalage au second nombre entier. Si une virgule est utilisée au lieu du mot-clé OFFSET, le décalage correspond au premier nombre et la limite au second nombre. Cette contradiction apparente est intentionnelle, car elle assure une compatibilité accrue avec les systèmes de bases de données SQL hérités.

Une instruction SELECT composée est constituée de plusieurs instructions SELECT simples connectées par l'un des opérateurs UNION, UNION ALL, INTERSECT ou EXCEPT. Dans une instruction SELECT composée, toutes les instructions SELECT constituantes doivent spécifier le même nombre de colonnes de résultats. Une seule clause ORDER BY doit suivre l'instruction SELECT finale (et précéder l'unique clause LIMIT si elle est spécifiée). Les opérateurs UNION et UNION ALL combinent les résultats des instructions SELECT précédentes et suivantes dans une même table. Avec l'opérateur UNION, toutes les lignes de résultat sont distinctes, alors qu'avec l'opérateur UNION ALL, il peut exister des doublons. L'opérateur INTERSECT prend l'intersection des résultats des instructions SELECT précédentes et suivantes. L'opérateur EXCEPT prend le résultat de l'instruction SELECT précédente après avoir supprimé les résultats de l'instruction SELECT suivante. Lorsque trois instructions SELECT ou plus sont connectées dans une instruction composée, elles sont groupées de la première à la dernière.

Pour consulter une définition des expressions autorisées, voir Expressions.

Depuis AIR 2.5, l'opérateur SQL CAST est pris en charge en lecture pour convertir des données BLOB en objets ActionScript ByteArray. Par exemple, le code suivant lit les données brutes qui ne sont pas stockées au format AMF et les stocke dans un objet ByteArray :

```
stmt.text = "SELECT CAST(data AS ByteArray) AS data FROM pictures;";
stmt.execute();
var result:SQLResult = stmt.getResult();
var bytes:ByteArray = result.data[0].data;
```

INSERT

L'instruction INSERT, dont il existe deux formes de base, permet d'entrer des données dans les tables.

```
sql-statement  ::=  INSERT [OR conflict-algorithm] INTO [database-name.] table-name [(column-
list)] VALUES (value-list) |
                    INSERT [OR conflict-algorithm] INTO [database-name.] table-name [(column-
list)] select-statement
               REPLACE INTO [database-name.] table-name [(column-list)] VALUES (value-list) |
                    REPLACE INTO [database-name.] table-name [(column-list)] select-statement
```

La première forme (qui contient le mot-clé VALUES) crée une ligne unique dans une table existante. Si aucun élément column-list n'est spécifié, le nombre de valeurs doit correspondre au nombre de colonnes de la table. Si un élément column-list est spécifié, le nombre de valeurs doit correspondre au nombre de colonnes spécifié. Les colonnes de la table qui n'apparaissent pas dans la liste de colonnes sont remplies avec les valeurs par défaut définies lors de la création de la table ou avec NULL si aucune valeur par défaut n'est définie.

La seconde forme de l'instruction INSERT extrait ses données d'une instruction SELECT. Le nombre de colonnes du résultat de l'instruction SELECT doit correspondre exactement au nombre de colonnes de la table si aucun élément column-list n'est spécifié, ou au nombre de colonnes nommées dans l'élément column-list. Une nouvelle entrée est créée dans la table pour chaque ligne du résultat de l'instruction SELECT. L'instruction SELECT peut être simple ou composée. Pour consulter la définition des instructions SELECT autorisées, voir SELECT.

Le paramètre conflict-algorithm facultatif permet de spécifier un autre algorithme de résolution de conflits de contraintes à utiliser pendant l'exécution de cette commande. Pour une explication et une définition des algorithmes de conflit, voir « Instructions et clauses spéciales » à la page 1166.

Les deux formes REPLACE INTO de l'instruction reviennent à utiliser la forme INSERT standard [OR conflict-algorithm] avec l'algorithme de conflit REPLACE (c'est-à-dire la forme INSERT OR REPLACE...).

Les deux formes REPLACE INTO de l'instruction reviennent à utiliser la forme INSERT standard [OR conflict-algorithm] avec l'algorithme de conflit REPLACE (c'est-à-dire la forme INSERT OR REPLACE...).

UPDATE

La commande de mise à jour modifie les enregistrements existants d'un tableau.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,...
[WHERE expr]
```

La commande se compose du mot-clé UPDATE, suivi du nom du tableau dont vous souhaitez mettre à jour les enregistrements. Après le mot-clé SET, saisissez le nom de la colonne et la valeur cible de la colonne, sous forme de liste de valeurs séparées par une virgule. L'expression de clause WHERE indique les lignes des enregistrements à mettre à jour.

DELETE

La commande delete permet de supprimer des enregistrements dans une table.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

Elle se compose des mots-clés DELETE FROM, suivis du nom de la table dans laquelle seront supprimés les enregistrements.

S'il n'existe pas de clause WHERE, toutes les lignes de la table sont supprimées. S'il existe une clause WHERE, seules les lignes qui correspondent à l'expression sont supprimées. L'expression de clause WHERE doit donner une valeur booléenne. Pour consulter une définition des expressions autorisées, voir Expressions.

Instructions de définition de données

Les instructions de définition des données permettent de créer, de modifier et de supprimer des objets de base de données, tels que des tables, des vues, des index et des déclencheurs. Les instructions de définition de données suivantes sont prises en charge :

- Tables :
  - CREATE TABLE
  - ALTER TABLE

    - DROP TABLE
- Index :
    - CREATE INDEX
    - DROP INDEX
- Vues :
    - CREATE VIEWS
    - DROP VIEWS
- Déclencheurs :
    - CREATE TRIGGERS
    - DROP TRIGGERS

CREATE TABLE

Une instruction CREATE TABLE se compose des mots-clés CREATE TABLE, suivis du nom de la nouvelle table, puis (entre parenthèses) d'une liste de contraintes et de définitions de colonne. Le nom de la table peut être un identificateur ou une chaîne.

```
sql-statement      ::=  CREATE [TEMP | TEMPORARY] TABLE [IF NOT EXISTS] [database-name.] table-
name
                          ( column-def [, column-def]* [, constraint]* )
sql-statement      ::=  CREATE [TEMP | TEMPORARY] TABLE [database-name.] table-name AS select-
statement
column-def          ::=  name [type] [[CONSTRAINT name] column-constraint]*
type                ::=  typename | typename ( number ) | typename ( number , number )
column-constraint   ::=  NOT NULL [ conflict-clause ] |
                         PRIMARY KEY [sort-order] [ conflict-clause ] [AUTOINCREMENT] |
                         UNIQUE [conflict-clause] |
                         CHECK ( expr ) |
                         DEFAULT default-value |
                         COLLATE collation-name
constraint          ::=  PRIMARY KEY ( column-list ) [conflict-clause] |
                         UNIQUE ( column-list ) [conflict-clause] |
                         CHECK ( expr )
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-algorithm  ::=  ROLLBACK | ABORT | FAIL | IGNORE | REPLACE
default-value     ::=  NULL | string | number | CURRENT_TIME | CURRENT_DATE | CURRENT_TIMESTAMP
sort-order          ::=  ASC | DESC
collation-name      ::=  BINARY | NOCASE
column-list         ::=  column-name [, column-name]*
```

Chaque définition de colonne se compose du nom de la colonne, suivi de son type de données, puis d'une ou de plusieurs contraintes de colonne facultatives. Le type de données de la colonne limite les données pouvant y être stockées. Si vous tentez de stocker une valeur dans une colonne dotée d'un type de données différent, le moteur d'exécution convertit cette valeur en une valeur du type approprié (le cas échéant) ou renvoie une erreur. Pour plus d'informations, voir la section Prise en charge des types de données.

La contrainte de colonne NOT NULL indique que la colonne ne doit pas contenir de valeurs NULL.

Une contrainte UNIQUE entraîne la création d'un index sur les colonnes spécifiées. Cet index doit contenir des clés uniques. Deux lignes ne peuvent pas contenir de valeurs ou de combinaisons de valeurs identiques pour les colonnes spécifiées. Une instruction CREATE TABLE peut comporter plusieurs contraintes UNIQUE, y compris plusieurs colonnes associées à une contrainte UNIQUE dans la définition des colonnes et/ou plusieurs contraintes UNIQUE au niveau de la table.

Une contrainte CHECK définit une expression qui est évaluée et doit être vraie pour que les données d'une ligne soient insérées ou mises à jour. L'expression CHECK doit donner une valeur booléenne.

Une clause COLLATE dans une définition de colonne spécifie la fonction de classement à utiliser pour comparer les entrées texte de la colonne. La fonction de classement BINARY est utilisée par défaut. Pour plus d'informations sur la clause COLLATE et les fonctions de classement, voir COLLATE.

Les contraintes DEFAULT spécifient la valeur par défaut à utiliser avec une instruction INSERT. La valeur peut être NULL, une constante de chaîne ou un nombre. La valeur par défaut peut également être l'un des mots-clés spéciaux indépendants de la casse CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP. Si la valeur est NULL, une constante de chaîne ou un nombre, elle est insérée littéralement dans la colonne à chaque fois qu'une instruction INSERT ne spécifie pas de valeur pour cette colonne. Si la valeur est CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP, la date UTC et/ou l'heure est insérée dans la colonne. CURRENT_TIME gère le format HH:MM:SS. CURRENT_DATE gère le format AAAA-MM-JJ. CURRENT_TIMESTAMP gère le format AAAA-MM-JJ HH:MM:SS.

La définition d'une contrainte PRIMARY KEY crée généralement un index UNIQUE sur les colonnes correspondantes. Toutefois, si la contrainte PRIMARY KEY s'applique à une seule colonne dont le type de données est INTEGER (ou l'un de ses synonymes, tels qu'int), cette colonne est alors utilisée par la base de données en tant que clé primaire réelle de la table. En d'autres termes, la colonne ne doit contenir que des valeurs entières uniques. (Notez que dans de nombreuses implémentations de SQLite, seul le type de colonne INTEGER entraîne l'utilisation de la colonne en tant que clé primaire interne. Néanmoins, dans Adobe AIR, les synonymes d'INTEGER, tels que int, spécifient également ce comportement.)

Si une table ne possède pas de colonne INTEGER PRIMARY KEY, une clé de type entier est automatiquement générée à chaque fois qu'une ligne est insérée. L'accès à la clé primaire d'une ligne s'effectue toujours par l'un des noms spéciaux ROWID, OID ou _ROWID_. Ces noms peuvent être utilisés qu'il s'agisse d'une valeur INTEGER PRIMARY KEY déclarée de façon explicite ou d'une valeur générée en interne. Toutefois, si la table contient une valeur INTEGER PRIMARY KEY explicite, le nom de la colonne dans les données de résultat correspond au nom de la colonne en tant que tel et non au nom spécial.

Une colonne INTEGER PRIMARY KEY peut également inclure le mot-clé AUTOINCREMENT. Si le mot-clé AUTOINCREMENT est utilisé, la base de données génère et insère automatiquement une clé de type entier incrémentée séquentiellement dans la colonne INTEGER PRIMARY KEY lorsqu'elle exécute une instruction INSERT qui ne spécifie aucune valeur explicite pour la colonne.

Une instruction CREATE TABLE ne peut comporter qu'une seule contrainte PRIMARY KEY. Elle peut faire partie de la définition d'une colonne ou d'une contrainte PRIMARY KEY unique au niveau de la table. Une colonne de clé primaire possède implicitement la valeur NOT NULL.

Le paramètre conflict-clause facultatif qui suit plusieurs contraintes permet de spécifier un autre algorithme de résolution de conflits de contraintes par défaut pour cette contrainte. La valeur par défaut est ABORT. Les différentes contraintes d'une même table peuvent gérer différents algorithmes de résolution de conflits par défaut. Si une instruction INSERT ou UPDATE spécifie un autre algorithme de résolution de conflits, celui-ci est utilisé à la place de l'algorithme spécifié dans l'instruction CREATE TABLE. Pour plus d'informations, voir la section ON CONFLICT sous « Instructions et clauses spéciales » à la page 1166.

Les contraintes complémentaires, telle FOREIGN KEY, ne génèrent pas d'erreur, mais sont ignorées par le moteur d'exécution.

Si le mot-clé TEMP ou TEMPORARY figure entre CREATE et TABLE, la table créée n'est alors visible que dans le cadre de la même connexion à la base de données (occurrence de SQLConnection). Elle est automatiquement supprimée lors de la fermeture de la connexion à la base de données. Tous les index associés à une table temporaire sont également temporaires. Les tables et index temporaires sont stockés dans un fichier distinct du fichier de base de données principal.

Si le préfixe database-name facultatif est spécifié, la table est alors créée dans une base de données nommée (la base de données connectée à l'occurrence de SQLConnection en appelant la méthode attach() avec le nom de base de données spécifié). Spécifier à la fois un préfixe database-name et le mot-clé TEMP est une erreur, sauf si le préfixe database-name est temp. Lorsqu'aucun nom de base de données n'est spécifié et que le mot-clé TEMP n'est pas présent, la table est créée dans la base de données principale (la base de données connectée à l'occurrence de SQLConnection par la méthode open() ou openAsync()).

Le nombre de colonnes ou de contraintes présentes dans une table n'est sujet à aucune limite arbitraire. Le volume de données présent dans une ligne n'est pas non plus limité.

La forme CREATE TABLE AS définit la table en tant que jeu de résultats d'une requête. Les noms des colonnes de la table sont identiques à ceux des colonnes de résultat.

Si la clause IF NOT EXISTS facultative est présente et qu'une autre table porte déjà le même nom, la base de données ignore alors la commande CREATE TABLE.

L'instruction DROP TABLE permet de supprimer une table et l'instruction ALTER TABLE d'apporter des modifications limitées.

ALTER TABLE

La commande ALTER TABLE permet à l'utilisateur de renommer une table existante ou de lui ajouter une nouvelle colonne. Il est impossible de supprimer une colonne dans une table.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

La syntaxe RENAME TO permet de renommer la table identifiée par [database-name.] table-name en new-table-name. Cette commande ne permet pas de déplacer une table entre les bases de données jointes, mais uniquement de renommer une table dans la même base de données.

Si la table renommée contient des déclencheurs ou des index, ils demeurent joints à la table après son changement de nom. Toutefois, si des définitions de vue ou des instructions sont exécutées par des déclencheurs faisant référence à la table renommée, elles ne sont pas modifiées automatiquement de sorte à utiliser le nouveau nom de la table. Si une table renommée est associée à des vues ou des déclencheurs, vous devez supprimer et recréer manuellement les déclencheurs ou les définitions de vue en utilisant le nouveau nom de la table.

La syntaxe ADD [COLUMN] permet d'ajouter une colonne à une table existante. La nouvelle colonne est toujours ajoutée à la fin de la liste de colonnes existantes. La clause column-def peut prendre toutes les formes autorisées dans une instruction CREATE TABLE, mais est sujette aux restrictions suivantes :

- La colonne ne doit pas contenir de contrainte PRIMARY KEY ou UNIQUE.
- La colonne ne doit pas posséder la valeur par défaut CURRENT_TIME, CURRENT_DATE ou CURRENT_TIMESTAMP.
- Si une contrainte NOT NULL est spécifiée, la colonne doit posséder une valeur par défaut autre que NULL.

La durée d'exécution de l'instruction ALTER TABLE n'est pas affectée par la quantité de données que contient la table.

DROP TABLE

L'instruction DROP TABLE supprime une table ajoutée à l'aide de l'instruction CREATE TABLE. C'est la table contenant le paramètre table-name spécifié qui est supprimée. Elle disparaît totalement de la base de donnée et du fichier sur disque. Il est impossible de restaurer la table. Tous les index associés à la table sont également supprimés.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

Par défaut, l'instruction DROP TABLE ne réduit pas la taille du fichier de base de données. L'espace vide de la base de données est conservé et exploité lors des opérations INSERT suivantes. Pour supprimer l'espace libre dans la base de données, utilisez la méthode SQLConnection.clean(). Si le paramètre autoClean est défini sur true lors de la création initiale de la base de données, l'espace est automatiquement libéré.

La clause IF EXISTS facultative supprime l'erreur qui se produit normalement si la table n'existe pas.

CREATE INDEX

La commande CREATE INDEX se compose des mots-clés CREATE INDEX, suivis du nom du nouvel index, du mot-clé ON, du nom de la table créée précédemment à indexer et d'une liste de noms de colonnes de la table (entre parenthèses) dont les valeurs sont utilisées pour la clé d'index.

```
sql-statement  ::=  CREATE [UNIQUE] INDEX [IF NOT EXISTS] [database-name.] index-name
                    ON table-name ( column-name [, column-name]* )
column-name    ::=  name [COLLATE collation-name] [ASC | DESC]
```

Chaque nom de colonne peut être suivi des mots-clés ASC ou DESC pour indiquer l'ordre de tri, mais cette indication est ignorée par le moteur d'exécution. Un tri est toujours exécuté par ordre croissant.

La clause COLLATE qui suit chaque nom de colonne définit une séquence de classement utilisée pour les valeurs texte de la colonne. La séquence de classement par défaut est définie pour cette colonne dans l'instruction CREATE TABLE. Si aucune séquence de classement n'est spécifiée, la séquence de classement BINARY est utilisée. Pour consulter une définition de la clause COLLATE et des fonctions de classement, voir COLLATE.

Le nombre d'index pouvant être associés à une même table n'est sujet à aucune restriction arbitraire. Le nombre de colonnes d'un index n'est pas non plus restreint.

DROP INDEX

L'instruction DROP INDEX supprime un index ajouté à l'aide de l'instruction CREATE INDEX. L'index spécifié est totalement supprimé du fichier de base de données. L'unique façon de restaurer l'index consiste à entrer à nouveau la commande CREATE INDEX appropriée.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

Par défaut, l'instruction DROP INDEX ne réduit pas la taille du fichier de base de données. L'espace vide de la base de données est conservé et exploité lors des opérations INSERT suivantes. Pour supprimer l'espace libre dans la base de données, utilisez la méthode SQLConnection.clean(). Si le paramètre autoClean est défini sur true lors de la création initiale de la base de données, l'espace est automatiquement libéré.

CREATE VIEW

La commande CREATE VIEW affecte un nom à une instruction SELECT prédéfinie. Ce nouveau nom peut ensuite être utilisé dans une clause FROM d'une autre instruction SELECT au lieu d'un nom de table. Les vues permettent généralement de simplifier les requêtes en combinant un ensemble de données complexe (et fréquemment utilisé) en une structure exploitable dans d'autres opérations.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS
select-statement
```

Si le mot-clé TEMP ou TEMPORARY figure entre CREATE et VIEW, la vue créée est réservée à l'occurrence de SQLConnection qui a ouvert la base de données et est automatiquement supprimée lors de la fermeture de cette dernière.

Si un élément [database-name] est spécifié, la table est créée dans la base de données nommée (une base de données connectée à l'occurrence de SQLConnection en appelant la méthode attach() avec l'argument name spécifié). Spécifier à la fois un élément [database-name] et le mot-clé TEMP est une erreur, sauf si [database-name] correspond à temp. Lorsqu'aucun nom de base de données n'est spécifié et que le mot-clé TEMP n'est pas présent, la vue est créée dans la base de données principale (la base de données connectée à l'occurrence de SQLConnection avec la méthode open() ou openAsync()).

Les vues sont disponibles en lecture seule. Il est impossible d'utiliser une instruction DELETE, INSERT ou UPDATE sur une vue, sauf si un déclencheur au moins du type associé (INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE) est défini. Pour plus d'informations sur la création d'un déclencheur associé à une vue, voir CREATE TRIGGER.

Pour supprimer une vue d'une base de données, utilisez l'instruction DROP VIEW.

DROP VIEW

L'instruction DROP VIEW supprime une vue créée par une instruction CREATE VIEW.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

Le paramètre view-name spécifié indique le nom de la vue à supprimer. Elle est supprimée de la base de données, mais aucune donnée figurant dans les tables sous-jacentes n'est modifiée.

CREATE TRIGGER

L'instruction CREATE TRIGGER permet d'ajouter des déclencheurs au schéma de la base de données. Un déclencheur est une opération de base de données (trigger-action) exécutée automatiquement lorsqu'un événement de base de données spécifié (database-event) se produit.

```
sql-statement   ::=  CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [database-name.] trigger-
name
                     [BEFORE | AFTER] database-event
                     ON table-name
                     trigger-action
sql-statement   ::=  CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [database-name.] trigger-
name
                     INSTEAD OF database-event
                     ON view-name
                     trigger-action
database-event  ::=  DELETE |
                     INSERT |
                     UPDATE |
                     UPDATE OF column-list
trigger-action  ::=  [FOR EACH ROW] [WHEN expr]
                     BEGIN
                       trigger-step ;
                       [ trigger-step ; ]*
                     END
trigger-step    ::=  update-statement |
                     insert-statement |
                     delete-statement |
                     select-statement
column-list     ::=  column-name [, column-name]*
```

L'exécution d'un déclencheur est spécifiée à chaque fois qu'une instruction DELETE, INSERT ou UPDATE se produit sur une table de base de données donnée, ou à chaque fois qu'une instruction UPDATE associée à une ou plusieurs colonnes spécifiées d'une table est implémentée. A moins d'utiliser le mot-clé TEMP ou TEMPORARY, les déclencheurs sont permanents. Si tel est le cas, le déclencheur est supprimé lors de la fermeture de la connexion à la base de données principale liée à l'occurrence de SQLConnection. Lorsqu'aucun repère temporel n'est spécifié (BEFORE or AFTER), le déclencheur est défini par défaut sur BEFORE.

Etant donné que seuls les déclencheurs FOR EACH ROW sont pris en charge, le texte FOR EACH ROW est facultatif. Avec un déclencheur FOR EACH ROW, les instructions trigger-step sont exécutées pour chaque ligne de base de données insérée, mise à jour ou supprimée par l'instruction qui entraîne l'exécution du déclencheur, si l'expression de la clause WHEN est true.

Si une clause WHEN est stipulée, les instructions SQL spécifiées en tant « qu'étapes du déclencheur » ne sont exécutées que pour les lignes pour lesquelles la clause WHEN est true. Si aucune clause WHEN n'est stipulée, les instructions SQL sont exécutées pour toutes les lignes.

Dans le corps d'un déclencheur (clause trigger-action), les valeurs de la table affectée avant et après la modification sont disponibles à l'aide des noms de table spéciaux OLD et NEW. La structure des tables OLD et NEW correspond à la structure de la table sur laquelle est créé le déclencheur. La table OLD contient les lignes modifiées ou supprimées par l'instruction de déclenchement, dans l'état qui précède les opérations de cette dernière. La table NEW contient les lignes modifiées ou supprimées par l'instruction de déclenchement, dans l'état qui suit les opérations de cette dernière. La clause WHEN et les instructions trigger-step peuvent accéder aux valeurs des lignes insérées, supprimées ou mises à jour à l'aide de références exprimées sous la forme NEW.column-name et OLD.column-name, où column-name est le nom d'une colonne de la table à laquelle est associé le déclencheur. La disponibilité des références de table OLD et NEW varie selon le type d'événement de base de données géré par le déclencheur :

- INSERT : les références NEW sont valides.
- UPDATE : les références NEW et OLD sont valides.
- DELETE : les références OLD sont valides.

Le repère temporel spécifié (BEFORE, AFTER ou INSTEAD OF) détermine le moment où les instructions trigger-step sont exécutées par rapport à l'insertion, la modification ou la suppression de la ligne associée. Une clause ON CONFLICT peut être spécifiée dans le cadre d'une instruction UPDATE ou INSERT dans une instruction trigger-step. Toutefois, si une clause ON CONFLICT est spécifiée dans le cadre de l'instruction responsable de l'exécution du déclencheur, la règle de gestion des conflits correspondante est alors utilisée.

Outre les déclencheurs de table, un déclencheur INSTEAD OF peut être associé à une vue. Si un ou plusieurs déclencheurs INSTEAD OF INSERT, INSTEAD OF DELETE ou INSTEAD OF UPDATE sont définis sur une vue, l'exécution du type associé d'instruction (INSERT, DELETE ou UPDATE) sur la vue est considérée comme correcte. Dans ce cas, l'exécution d'une instruction INSERT, DELETE ou UPDATE sur la vue entraîne l'activation des déclencheurs associés. Etant donné qu'il s'agit d'un déclencheur INSTEAD OF, les tables sous-jacentes de la vue ne sont pas modifiées par l'instruction qui entraîne l'activation du déclencheur. Les déclencheurs permettent cependant d'exécuter des opérations de modification sur les tables sous-jacentes.

Lors de la création d'un déclencheur sur une table contenant une colonne INTEGER PRIMARY KEY, il est important de ne pas oublier l'élément suivant. Si un déclencheur BEFORE modifie la colonne INTEGER PRIMARY KEY d'une ligne qui doit être mise à jour par l'instruction responsable de l'activation du déclencheur, la mise à jour ne se produit pas. La solution consiste à créer la table avec une colonne PRIMARY KEY au lieu d'une colonne INTEGER PRIMARY KEY.

L'instruction DROP TRIGGER permet de supprimer un déclencheur. Lors de la suppression d'une table ou d'une vue, tous les déclencheurs associés à la table ou à la vue sont automatiquement supprimés.

Fonction RAISE ()

Une fonction SQL spéciale RAISE() peut être utilisée dans une instruction trigger-step de déclencheur. La syntaxe de cette fonction est la suivante :

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

Lorsque l'une des trois premières formes est appelée pendant l'exécution du déclencheur, l'action de traitement ON CONFLICT spécifiée (ABORT, FAIL ou ROLLBACK) est exécutée et l'exécution de l'instruction en cours se termine. L'action ROLLBACK étant considérée comme un échec de l'exécution de l'instruction, l'occurrence de SQLStatement dont la méthode execute() était exécutée distribue un événement d'erreur (SQLErrorEvent.ERROR). L'objet SQLError de la propriété error de l'objet d'événement distribué voit sa propriété details définie sur le message d'erreur spécifié par la fonction RAISE().

Lors de l'appel de la fonction RAISE(IGNORE), la suite du déclencheur en cours, ainsi que l'instruction responsable de son activation et les déclencheurs éventuellement exécutés par la suite sont abandonnés. Les modifications apportées à la base de données ne sont pas annulées. Si l'instruction responsable de l'exécution du déclencheur fait elle-même partie d'un déclencheur, ce programme est réactivé au début de l'étape suivante. Pour plus d'informations sur les algorithmes de résolution de conflit, voir la section ON CONFLICT (algorithmes de conflit).

DROP TRIGGER

L'instruction DROP TRIGGER supprime un déclencheur créé par l'instruction CREATE TRIGGER.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

Le déclencheur est supprimé de la base de données. Notez que les déclencheurs sont automatiquement supprimés lors de la suppression de la table associée.

Instructions et clauses spéciales

Cette section décrit plusieurs clauses qui sont des extensions de SQL fournies par le moteur d'exécution, ainsi que deux éléments de langage utilisables dans de nombreux commentaires, instructions et expressions.

COLLATE

La clause COLLATE est utilisée dans les instructions SELECT, CREATE TABLE et CREATE INDEX pour définir l'algorithme de comparaison qui permet de comparer et trier des valeurs.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

Le type de classement par défaut des colonnes est BINARY. Lorsque le classement BINARY est utilisé avec des valeurs de la classe de stockage TEXT, un classement binaire est effectué en comparant les octets en mémoire qui représentent la valeur, quel que soit le codage du texte.

La séquence de classement NOCASE n'est appliquée qu'aux valeurs de la classe de stockage TEXT. Lorsqu'il est utilisé, le classement NOCASE effectue une comparaison sans respecter la casse.

Aucune séquence de classement n'est utilisée pour les classes de stockage de type NULL, BLOB, INTEGER ou REAL.

Pour utiliser un type de classement autre que BINARY avec une colonne, une clause COLLATE doit être spécifiée dans le cadre de la définition de la colonne dans l'instruction CREATE TABLE. Chaque fois que deux valeurs TEXT sont comparées, une séquence de classement permet de déterminer les résultats de la comparaison, selon les règles suivantes :

- Pour les opérateurs de comparaison binaire, si l'un des deux opérandes est une colonne, le type de classement par défaut de la colonne détermine la séquence de classement utilisée pour la comparaison. Si les deux opérandes sont des colonnes, le type de classement de l'opérande de gauche détermine la séquence de classement utilisée. Si aucun des opérandes n'est une colonne, la séquence de classement BINARY est utilisée.
- L'opérateur BETWEEN...AND revient à utiliser deux expressions avec les opérateurs >= et <=. Par exemple, l'expression x BETWEEN y AND z est équivalente à x >= y AND x <= z. Par conséquent, l'opérateur BETWEEN...AND suit la règle précédente pour identifier la séquence de classement.
- L'opérateur IN se comporte comme l'opérateur = pour déterminer la séquence de classement à utiliser. Par exemple, la séquence de classement utilisée pour l'expression IN (y, z) est le type de classement par défaut de x si x est une colonne. Dans le cas contraire, le classement BINARY est utilisé.
- Une clause ORDER BY faisant partie d'une instruction SELECT peut se voir affecter explicitement une séquence de classement à utiliser pour l'opération de tri. Dans ce cas, la séquence de classement explicite est toujours utilisée. Si tel n'est pas le cas, lorsque l'expression triée par une clause ORDER BY est une colonne, le type de classement par défaut de cette colonne est utilisé pour déterminer l'ordre de tri. Si l'expression n'est pas une colonne, la séquence de classement BINARY est utilisée.

EXPLAIN

Le modificateur de commande EXPLAIN est une extension non standard de SQL.

```
sql-statement  ::=  EXPLAIN sql-statement
```

Si le mot-clé EXPLAIN apparaît avant une autre instruction SQL, au lieu d'exécuter la commande, le résultat signale la séquence des instructions d'un ordinateur virtuel qu'il aurait utilisée pour exécuter la commande si le mot-clé EXPLAIN n'avait pas été présent. EXPLAIN est une fonctionnalité avancée qui permet aux développeurs de modifier le texte d'une instruction SQL dans le but d'optimiser les performances ou de déboguer une instruction qui ne semble pas fonctionner correctement.

ON CONFLICT (algorithmes de conflit)

La clause ON CONFLICT n'est pas une commande SQL distincte, mais une clause non standard susceptible de figurer dans un grand nombre d'autres commandes SQL.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

La première forme de la clause ON CONFLICT, qui fait appel à des mots-clés ON CONFLICT, permet de créer une instruction CREATE TABLE. Pour une instruction INSERT ou UPDATE, la seconde forme est utilisée, ON CONFLICT étant remplacé par OR pour rendre la syntaxe plus naturelle. Ainsi, au lieu d'INSERT ON CONFLICT IGNORE, l'instruction devient INSERT OR IGNORE. Bien que les mots-clés soient différents, la signification de la clause est identique pour les deux formes.

La clause ON CONFLICT définit l'algorithme requis pour résoudre les conflits de contraintes. Les algorithmes sont au nombre de cinq, ROLLBACK, ABORT, FAIL, IGNORE et REPLACE. L'algorithme par défaut est ABORT. Les cinq algorithmes de conflit sont expliqués ci-après :

ROLLBACK Lorsqu'une violation de contrainte se produit, ROLLBACK survient immédiatement et met fin à la transaction en cours. La commande est annulée et l'occurrence de SQLStatement distribue un événement d'erreur. Si aucune transaction n'est active (mise à part la transaction implicite créée pour chaque commande), cet algorithme fonctionne comme ABORT.

ABORT Lorsqu'une violation de contrainte se produit, la commande retire toute modification précédente éventuellement exécutée et l'occurrence de SQLStatement distribue un événement d'erreur. Aucun algorithme ROLLBACK n'étant exécuté, les modifications apportées par les commandes précédentes dans une transaction sont donc conservées. ABORT correspond au comportement par défaut.

FAIL Lorsqu'une violation de contrainte se produit, la commande est annulée et l'occurrence de SQLStatement distribue un événement d'erreur. Toutefois, toutes les modifications apportées à la base de données par l'instruction avant l'identification de la violation de contrainte sont conservées et ne sont pas retirées. Par exemple, si une instruction UPDATE rencontre une violation de contrainte au niveau de la centième ligne lors de la tentative de mise à jour, les modifications apportées aux 99 premières lignes sont conservées, mais la ligne 100 et les suivantes ne sont pas modifiées.

IGNORE Lorsqu'une violation de contrainte se produit, la ligne concernée n'est ni insérée, ni modifiée. Mise à part la non -prise en compte de cette ligne, la commande poursuit son exécution normalement. L'insertion ou la mise à jour des lignes qui précèdent et qui suivent la ligne contenant la violation de contrainte se poursuit normalement. Aucune erreur n'est renvoyée.

REPLACE Lorsqu'une violation de contrainte UNIQUE se produit, les lignes préexistantes à l'origine de cette violation sont supprimées avant l'insertion ou la mise à jour de la ligne en cours. Par conséquent, l'insertion ou la mise à jour se produit toujours, et l'exécution de la commande se poursuit normalement. Aucune erreur n'est renvoyée. Si une violation de contrainte NOT NULL se produit, la valeur NULL est remplacée par la valeur par défaut de la colonne. Si la colonne n'a pas de valeur par défaut, l'algorithme ABORT est utilisé. Si une violation de contrainte CHECK se produit, l'algorithme IGNORE est utilisé. Lorsque cette stratégie de résolution de conflits supprime des lignes pour respecter une contrainte, elle n'invoque pas de déclencheurs de suppression sur ces lignes.

L'algorithme spécifié dans la clause OR d'une instruction INSERT ou UPDATE remplace tout algorithme spécifié dans une instruction CREATE TABLE. Si aucun algorithme n'est spécifié dans l'instruction CREATE TABLE ou dans l'instruction INSERT ou UPDATE en cours d'exécution, l'algorithme ABORT est utilisé.

REINDEX

La commande REINDEX permet de supprimer et de recréer un ou plusieurs index. Elle s'avère utile lorsque la définition d'une séquence de classement a été modifiée.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

Dans la première forme, tous les index de toutes les bases de données jointes qui utilisent la séquence de classement nommée sont recréés. Dans la seconde forme, lorsqu'un paramètre table-name est spécifié, tous les index associés à la table sont reconstruits. Si un paramètre index-name est spécifié, seul l'index spécifié est supprimé et recréé.

COMMENTAIRES

Les commentaires ne sont pas des commandes SQL, mais ils peuvent survenir dans des requêtes SQL. Ils sont traités comme des espaces blancs par le moteur d'exécution. Ils peuvent débuter à chaque fois qu'un espace blanc est détecté, y compris à l'intérieur d'expressions réparties sur plusieurs lignes.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

Un commentaire d'une seule ligne est indiqué par deux tirets. Un commentaire d'une seule ligne ne dépasse pas la fin de la ligne en cours.

Les blocs de commentaires peuvent s'étendre sur un nombre illimité de lignes ou être intégrés à une seule ligne. Si aucun délimiteur de terminaison n'a été défini, un bloc de commentaires atteint la fin de l'entrée. Cette situation n'est pas traitée comme une erreur. Une nouvelle instruction SQL peut débuter sur une ligne après la fin d'un bloc de commentaires. Les blocs de commentaires peuvent être intégrés à tout emplacement susceptible d'être occupé par un espace blanc, y compris dans des expressions et au milieu d'autres instructions SQL. Les blocs de commentaires ne s'imbriquent pas. Les commentaires d'une seule ligne placés dans un bloc de commentaires sont ignorés.

EXPRESSIONS

Les expressions sont des sous-commandes placées dans d'autres blocs SQL. La syntaxe valide d'une expression placée dans une instruction SQL est la suivante :

```
expr             ::=  expr binary-op expr |
                      expr [NOT] like-op expr [ESCAPE expr] |
                      unary-op expr |
                      ( expr ) |
                      column-name |
                      table-name.column-name |
                      database-name.table-name.column-name |
                      literal-value |
                      parameter |
                      function-name( expr-list | * ) |
                      expr ISNULL |
                      expr NOTNULL |
                      expr [NOT] BETWEEN expr AND expr |
                      expr [NOT] IN ( value-list ) |
                      expr [NOT] IN ( select-statement ) |
                      expr [NOT] IN [database-name.] table-name |
                      [EXISTS] ( select-statement ) |
                      CASE [expr] ( WHEN expr THEN expr )+ [ELSE expr] END |
                      CAST ( expr AS type ) |
                      expr COLLATE collation-name
like-op          ::=  LIKE | GLOB
binary-op        ::=  see Operators
unary-op         ::=  see Operators
parameter        ::=  :param-name | @param-name | ?
value-list       ::=  literal-value [, literal-value]*
literal-value    ::=  literal-string | literal-number | literal-boolean | literal-blob |
literal-null
literal-string   ::=  'string value'
literal-number   ::=  integer | number
literal-boolean   ::=  true | false
literal-blob   ::=  X'string of hexadecimal data'
literal-null   ::=  NULL
```

Une expression correspond à toute combinaison de valeurs et d'opérateurs pouvant être résolue en tant que valeur unique. Les expressions se divisent en deux types généraux, selon qu'elles donnent une valeur booléenne (true ou false) ou non.

Dans nombre de situations courantes, y compris dans une clause WHERE, une clause HAVING, l'expression ON dans une clause JOIN et une expression CHECK, l'expression doit donner une valeur booléenne. Les types d'expressions suivants remplissent cette condition :

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- Certaines fonctions
- Certains opérateurs (spécifiquement les opérateurs de comparaison)

Valeurs littérales

Une valeur numérique littérale est écrite sous forme de nombre entier ou de nombre à virgule flottante. La notation scientifique est prise en charge. Le caractère . (point) fait toujours office de caractère décimal.

Pour indiquer une chaîne littérale, placez-la entre guillemets droits simples (’). Pour inclure un guillemet droit simple dans une chaîne, insérez successivement deux guillemets droits simples, comme suit : ’’.

Une valeur booléenne littérale est indiquée par la valeur true ou false. Les valeurs booléennes littérales sont utilisées avec le type de données de colonne booléen.

Un littéral BLOB est un littéral de chaîne contenant des données hexadécimales et précédé d'un seul caractère x ou X, tel que X'53514697465'.

Une valeur littérale peut également être le jeton NULL.

Nom de colonne

Un nom de colonne peut être tout nom défini dans l'instruction CREATE TABLE ou l'un des identifiants spéciaux suivants : ROWID, OID ou _ROWID_. Ces identifiants spéciaux décrivent tous la clé aléatoire unique de type entier (« clé de la ligne »), associée à chaque ligne de chaque table. Les identifiants spéciaux ne font référence à la clé de la ligne que si l'instruction CREATE TABLE ne définit pas de colonne réelle portant le même nom. Les clés des lignes se comportent comme des colonnes en lecture seule. Une clé de ligne peut se substituer à toute colonne ordinaire, mais vous ne pouvez pas en modifier la valeur dans une instruction UPDATE ou INSERT. Le jeu de résultats de l'instruction de table SELECT * FROM ne comprend pas de clé de ligne.

Instruction SELECT

Une instruction SELECT peut apparaître dans une expression comme opérande de droite de l'opérateur IN, comme quantité scalaire (valeur de résultat unique) ou comme opérande d'un opérateur EXISTS. Lorsqu'elle est utilisée en tant que quantité scalaire ou qu'opérande d'un opérateur IN, le résultat de l'instruction SELECT ne peut contenir qu'une seule colonne. Une instruction SELECT composée (connectée par des mots-clés tels qu'UNION ou EXCEPT) est autorisée. Avec l'opérateur EXISTS, les colonnes du jeu de résultats de SELECT sont ignorées et l'expression renvoie TRUE si le jeu de résultats contient une ou plusieurs lignes, FALSE s'il est vide. Lorsqu'aucun terme de l'expression SELECT ne fait référence à la valeur de la requête conteneur, l'expression est évaluée une fois avant tout autre traitement et le résultat est réutilisé selon les besoins. Si l'expression SELECT contient des variables issues de la requête externe, appelée sous-requête corrélée, l'instruction SELECT est réévaluée à chaque fois que cela est nécessaire.

Lorsqu'une expression SELECT est l'opérande de droite de l'opérateur IN, l'opérateur IN renvoie TRUE si le résultat de l'opérande de gauche est égal à l'une des valeurs du jeu de résultats de l'instruction SELECT. L'opérateur IN peut être précédé du mot-clé NOT pour inverser le sens du test.

Lorsqu'une instruction SELECT apparaît dans une expression, mais n'est pas l'opérande droit d'un opérateur IN, la première ligne du résultat de SELECT devient la valeur utilisée dans l'expression. Si l'instruction SELECT produit plusieurs lignes de résultat, seule la première ligne est prise en compte. Si l'instruction SELECT ne produit pas de ligne de résultats, sa valeur est NULL.

Expression CAST

Une expression CAST remplace le type de données de la valeur spécifiée par le type donné. Le type spécifié peut être tout nom de type non vide géré par le type dans la définition de colonne d'une instruction CREATE TABLE. Pour plus d'informations, voir Prise en charge des types de données.

Autres éléments d'une expression

Vous pouvez également utiliser les éléments SQL suivants dans une expression :

- Fonctions intégrées : fonctions d'agrégation, fonctions scalaires et fonctions de formatage de la date et de l'heure
- Opérateurs
- Paramètres

Fonctions intégrées

Les fonctions intégrées sont divisées en trois catégories principales :

- Fonctions d'agrégation
- Fonctions scalaires
- Fonctions de date et d'heure

Outre ces fonctions, la fonction spéciale RAISE() permet de signaler une erreur lors de l'exécution d'un déclencheur. Cette fonction ne peut être utilisée que dans le corps d'une instruction CREATE TRIGGER. Pour plus d'informations sur la fonction RAISE(), voir CREATE TRIGGER > RAISE().

Comme tous les mots-clés dans SQL, les noms de fonction ne respectent pas la casse.

Fonctions d'agrégation

Les fonctions d'agrégation exécutent des opérations sur des valeurs issues de plusieurs lignes. Elles sont principalement utilisées dans des instructions SELECT en conjonction avec la clause GROUP BY.

AVG(X)	Renvoie la valeur moyenne de tous les X non NULL d'un groupe. Les valeurs de chaîne et les valeurs BLOB qui ne ressemblent pas à des nombres sont interprétées comme des 0. Le résultat d'AVG() est toujours une valeur à virgule flottante, même si toutes les entrées sont des nombres entiers.
COUNT(X) COUNT(*)	La première forme renvoie le nombre de fois que X n'est pas NULL dans un groupe. La seconde forme (associée à l'argument *) renvoie le nombre total de lignes du groupe.
MAX(X)	Renvoie la valeur maximale de toutes les valeurs du groupe. Pour déterminer cette valeur maximale, l'ordre de tri standard est utilisé.
MIN(X)	Renvoie la valeur minimale non NULL de toutes les valeurs du groupe. Pour déterminer cette valeur minimale, l'ordre de tri standard est utilisé. Si toutes les valeurs du groupe sont NULL, NULL est renvoyé.
SUM(X) TOTAL(X)	Renvoie la somme numérique de toutes les valeurs non NULL du groupe. Si toutes les valeurs sont NULL, SUM() renvoie NULL et TOTAL() renvoie 0.0. Le résultat de TOTAL() est toujours une valeur à virgule flottante. SUM() a pour résultat une valeur entière si toutes les entrées non NULL sont des nombres entiers. Si l'une des entrées de SUM() n'est pas un nombre entier et n'est pas NULL, SUM() renvoie une valeur à virgule flottante. Cette valeur peut être une approximation de la somme réelle.

Dans toutes les fonctions d'agrégation précédentes qui gèrent un seul argument, celui-ci peut être précédé du mot-clé DISTINCT. Dans ce cas, les éléments en double sont filtrés avant d'être transmis à la fonction d'agrégation. Par exemple, l'appel de la fonction COUNT(DISTINCT x) renvoie le nombre de valeurs distinctes de la colonne X au lieu du nombre total de valeurs non NULL que contient la colonne x.

Fonctions scalaires

Les fonctions scalaires opèrent sur les valeurs d'une ligne à la fois.

ABS(X)	Renvoie la valeur absolue de l'argument X.
COALESCE(X, Y, ...)	Renvoie une copie du premier argument non NULL. Si tous les arguments sont NULL, NULL est renvoyé. Deux arguments au moins doivent être définis.
GLOB(X, Y)	Cette fonction permet de mettre en œuvre la syntaxe X GLOB Y.

IFNULL(X, Y)	Renvoie une copie du premier argument non NULL. Si les deux arguments sont NULL, NULL est renvoyé. Cette fonction se comporte comme COALESCE().
HEX(X)	L'argument est interprété en tant que valeur de type de stockage BLOB. Le résultat est un rendu hexadécimal du contenu de cette valeur.
LAST_INSERT_ROWID( )	Renvoie l'identifiant de ligne (clé primaire générée) de la dernière ligne insérée dans la base de données via l'occurrence de SQLConnection en cours. Cette valeur est identique à la valeur renvoyée par la propriété SQLConnection.lastInsertRowID.
LENGTH(X)	Renvoie la longueur de la chaîne de X en caractères.
LIKE(X, Y [, Z])	Cette fonction permet de mettre en œuvre la syntaxe X LIKE Y [ESCAPE Z] de SQL. Si la clause ESCAPE facultative est présente, la fonction est appelée avec trois arguments. Dans le cas contraire, elle est appelée avec deux arguments seulement.
LOWER(X)	Renvoie une copie de la chaîne X dont tous les caractères sont convertis en minuscules.
LTRIM(X) LTRIM(X, Y)	Renvoie une chaîne qui résulte de la suppression des espaces du côté gauche de X. Si un argument Y est spécifié, la fonction supprime tous les caractères de Y du côté gauche de X.
MAX(X, Y, ...)	Renvoie l'argument possédant la valeur maximale. Outre les nombres, les arguments peuvent être des chaînes. La valeur maximale est déterminée par l'ordre de tri défini. Notez que MAX() est une fonction simple lorsqu'elle possède plusieurs arguments, mais une fonction d'agrégation lorsqu'elle n'en a qu'un seul.
MIN(X, Y, ...)	Renvoie l'argument possédant la valeur minimale. Outre les nombres, les arguments peuvent être des chaînes. La valeur minimale est déterminée par l'ordre de tri défini. Notez que MIN() est une fonction simple lorsqu'elle possède plusieurs arguments, mais une fonction d'agrégation lorsqu'elle n'en a qu'un seul.
NULLIF(X, Y)	Renvoie le premier argument si les arguments diffèrent, sinon elle renvoie NULL.
QUOTE(X)	Cette routine renvoie une chaîne correspondant à la valeur de l'argument pouvant être inclus dans une autre instruction SQL. Les chaînes sont entourées de guillemets simples avec caractères d'échappement sur les guillemets intérieurs si nécessaire. Les classes de stockage BLOB sont codées comme des littéraux hexadécimaux. La fonction s'avère utile lors de l'écriture de déclencheurs pour implémenter la fonctionnalité d'annulation et de rétablissement.
RANDOM(*)	Renvoie un nombre entier pseudo-aléatoire compris entre -9223372036854775808 et 9223372036854775807. Cette valeur aléatoire n'est pas chiffrée en dur.
RANDOMBLOB(N)	Renvoie un BLOB N-octets contenant des octets pseudo-aléatoires. N doit être un nombre entier positif. Cette valeur aléatoire n'est pas chiffrée en dur. Si la valeur de N est négative, un octet unique est renvoyé.
ROUND(X) ROUND(X, Y)	Arrondit le nombre X à Y chiffres après la virgule. Si l'argument Y est omis, 0 est utilisé.
RTRIM(X) RTRIM(X, Y)	Renvoie une chaîne qui résulte de la suppression des espaces du côté droit de X. Si un argument Y est spécifié, la fonction supprime tous les caractères de Y du côté droit de X.
SUBSTR(X, Y, Z)	Renvoie une sous-chaîne de la chaîne d'entrée X commençant par le Yième caractère et longue de Z caractères. Le caractère le plus à gauche de X est la position d'index 1. Si Y est négatif, le premier caractère de la sous-chaîne est détecté en comptant à partir de la droite et non à partir de la gauche.
TRIM(X) TRIM(X, Y)	Renvoie une chaîne qui résulte de la suppression des espaces du côté droit de X. Si un argument Y est spécifié, la fonction supprime tous les caractères de Y du côté droit de X.
TYPEOF(X)	Renvoie le type de l'expression X. Les valeurs gérées sont « null », « integer », « real », « text » et « blob ». Pour plus d'informations sur les types de données, voir Prise en charge des types de données.
UPPER(X)	Renvoie une copie de la chaîne d'entrée X convertie en majuscules.
ZEROBLOB(N)	Renvoie un BLOB contenant N octets de 0x00.

Fonctions de formatage de la date et de l'heure

Les fonctions de formatage de la date et de l'heure sont un groupe de fonctions scalaires qui permettent de créer des données de date et d'heure formatées. Notez que ces fonctions traitent et renvoient des valeurs de chaîne et des valeurs numériques. Elles ne sont pas conçues pour être utilisées avec le type de données DATE. Si vous utilisez ces fonctions sur les données d'une colonne dont le type de données déclaré est DATE, leur comportement est imprévisible.

DATE(T, ...)	La fonction DATE() renvoie une chaîne contenant la date au format AAAA-MM-JJ. Le premier paramètre (T) spécifie une chaîne horaire au format indiqué à la section Formats horaires. Vous pouvez spécifier un nombre illimité de modificateurs après la chaîne horaire. Les modificateurs sont recensés à la section Modificateurs.

TIME(T, ...)	La fonction TIME() renvoie une chaîne contenant l'heure exprimée au format HH:MM:SS. Le premier paramètre (T) spécifie une chaîne horaire au format indiqué à la section Formats horaires. Vous pouvez spécifier un nombre illimité de modificateurs après la chaîne horaire. Les modificateurs sont recensés à la section Modificateurs.
DATETIME(T, ...)	La fonction DATETIME() renvoie une chaîne contenant la date et l'heure au format AAAA-MM-JJ HH:MM:SS. Le premier paramètre (T) spécifie une chaîne horaire au format indiqué à la section Formats horaires. Vous pouvez spécifier un nombre illimité de modificateurs après la chaîne horaire. Les modificateurs sont recensés à la section Modificateurs.
JULIANDAY(T, ...)	La fonction JULIANDAY() renvoie un nombre qui indique le nombre de jours écoulés depuis midi heure de Greenwich le 24 novembre 4714 av. J.-C. et la date stipulée. Le premier paramètre (T) spécifie une chaîne horaire au format indiqué à la section Formats horaires. Vous pouvez spécifier un nombre illimité de modificateurs après la chaîne horaire. Les modificateurs sont recensés à la section Modificateurs.
STRFTIME(F, T, ...)	La routine STRFTIME() renvoie la date formatée selon la chaîne de format spécifiée en tant que premier argument F. La chaîne de format prend en charge les substitutions suivantes : %d : jour du mois %f : secondes fractionnaires (SS.SSS) %H : heure (00-24) %j : jour de l'année (001-366) %J : nombre de jours Julien %m : mois (01-12) %M : minute (00-59) %s : secondes depuis le 01-01-1970 %S : secondes (00-59) %w : jour de la semaine (0-6, dimanche = 0) %W : semaine de l'année (00-53) %Y : année (0000-9999) %% - % Le second paramètre (T) spécifie une chaîne horaire au format indiqué à la section Formats horaires. Vous pouvez spécifier un nombre illimité de modificateurs après la chaîne horaire. Les modificateurs sont recensés à la section Modificateurs.

Formats horaires

Une chaîne horaire gère les formats suivants :

AAAA-MM-JJ	2007-06-15
AAAA-MM-JJ HH:MM	2007-06-15 07:30
AAAA-MM-JJ HH:MM:SS	2007-06-15 07:30:59
AAAA-MM-JJ HH:MM:SS.SSS	2007-06-15 07:30:59.152
AAAA-MM-JJTHH:MM	2007-06-15T07:30
AAAA-MM-JJTHH:MM:SS	2007-06-15T07:30:59
AAAA-MM-JJTHH:MM:SS.SSS	2007-06-15T07:30:59.152
HH:MM	07:30 (la date est le 01-01-2000)
HH:MM:SS	07:30:59 (la date est le 01-01-2000)
HH:MM:SS.SSS	07:30:59:152 (la date est le 01-01-2000)
now	Date et heure en cours exprimées au format UCT (Universal Coordinated Time)
DDDD.DDDD	Nombre de jours Julien exprimé sous forme de nombre à virgule flottante.

Le caractère T figurant dans ces formats est un caractère littéral « T » qui sépare la date et l'heure. Les formats qui ne contiennent qu'une heure partent du principe que la date correspond au 01-01-2001.

Modificateurs

La chaîne horaire peut être suivie d'aucun ou de plusieurs modificateurs qui modifient la date ou en changent l'interprétation. Les modificateurs disponibles sont les suivants :

NNN days	Nombre de jours à ajouter à l'heure.
NNN hours	Nombre d'heures à ajouter à l'heure.
NNN minutes	Nombre de minutes à ajouter à l'heure.
NNN.NNNN seconds	Nombre de secondes et de millisecondes à ajouter à l'heure.
NNN months	Nombre de mois à ajouter à l'heure.
NNN years	Nombre d'années à ajouter à l'heure.
start of month	Ramène l'heure au début du mois.
start of year	Ramène l'heure au début de l'année.
start of day	Ramène l'heure au début de la journée.
weekday N	Fait avancer l'heure jusqu'au jour de la semaine spécifié. (0 = dimanche, 1 = lundi, etc.).
localtime	Convertit la date au format local.
utc	Convertit la date au format UCT (Universal Coordinated Time).

Opérateurs

SQL prend en charge un grand nombre d'opérateurs, dont les opérateurs courants qui existent dans la plupart des langages de programmation, ainsi que plusieurs opérateurs propres à SQL.

Opérateurs courants

Les opérateurs binaires suivants sont autorisés dans un bloc SQL et sont recensés par ordre de priorité décroissant :

```
*     /     %
+     -
<< >> & |
< >=    > >=
=     ==    !=    <> IN
AND
OR
```

Opérateurs de préfixe unaires pris en charge :

```
 !     ~     NOT
```

L'opérateur COLLATE peut être considéré comme un opérateur de suffixe unaire. Il possède la priorité la plus élevée. Il établit toujours une liaison plus étroite que tout opérateur de préfixe unaire ou opérateur binaire.

Notez qu'il existe deux variantes des opérateurs égal et différent. L'opérateur égal peut être = ou ==. L'opérateur différent peut être != ou <>.

L'opérateur || est l'opérateur de concaténation de chaînes. Il relie les deux chaînes de ses opérandes.

L'opérateur % a pour résultat le reste de son opérande de gauche modulo son opérande de droite.

Le résultat de tout opérateur binaire est une valeur numérique, à l'exception de l'opérateur de concaténation ||, dont le résultat est une chaîne.

Opérateurs SQL

LIKE

L'opérateur LIKE effectue une comparaison de correspondance basée sur un modèle.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

L'opérande de droite de l'opérateur LIKE contient le modèle, tandis que l'opérande de gauche contient la chaîne à comparer au modèle. Un symbole de pourcentage (%) dans le modèle représente un caractère générique. Il correspond à toute séquence de zéro, un ou plusieurs caractères de la chaîne. Un caractère de soulignement (_) dans le modèle correspond à tout caractère unique de la chaîne. Tous les autres caractères correspondent à leur valeur ou à leur équivalent majuscule/minuscule. En d'autres termes, la correspondance ne respecte pas la casse. (Remarque : le moteur de base de données gère la casse des caractères latins 7 bits uniquement. Par conséquent, l'opérateur LIKE respecte la casse des caractères UTF-8 ou iso8859 8 bits. Ainsi, l'expression 'a' LIKE 'A' est TRUE, mais 'æ' LIKE 'Æ' est FALSE). La propriété SQLConnection.caseSensitiveLike permet de modifier le respect de la casse pour les caractères latins.

Si la clause facultative ESCAPE est présente, l'expression qui suit le mot-clé ESCAPE doit correspondre à une chaîne composée d'un seul caractère. Ce caractère peut être utilisé dans le modèle LIKE pour correspondre aux caractères littéraux de pourcentage ou de soulignement. Le caractère échappement suivi du symbole de pourcentage, d'un soulignement ou de lui-même correspond respectivement à un symbole de pourcentage, à un soulignement ou à un caractère d'échappement littéral dans la chaîne.

GLOB

L'opérateur GLOB est similaire à LIKE, mais utilise la syntaxe globbing de fichier Unix pour ses caractères génériques. A l'encontre de LIKE, GLOB respecte la casse.

IN

L'opérateur IN calcule si son opérande de gauche est égal à l'une des valeurs de son opérande de droite (ensemble de valeurs entre parenthèses).

```
in-expr         ::=  expr [NOT] IN ( value-list ) |
                     expr [NOT] IN ( select-statement ) |
                     expr [NOT] IN [database-name.] table-name
value-list      ::=  literal-value [, literal-value]*
```

L'opérande de droite peut être un ensemble de valeurs littérales séparées par des virgules ou le résultat d'une instruction SELECT. Pour plus d'informations et pour connaître les restrictions d'utilisation d'une instruction SELECT en tant qu'opérande de droite de l'opérateur IN, voir les instructions SELECT dans les expressions.

BETWEEN...AND

L'opérateur BETWEEN...AND revient à utiliser deux expressions avec les opérateurs >= et <=. Par exemple, l'expression x BETWEEN y AND z est équivalente à x >= y AND x <= z.

NOT

L'opérateur NOT est un opérateur de négation. Les opérateurs GLOB, LIKE et IN peuvent être précédés du mot-clé NOT pour inverser le sens du test (en d'autres termes, pour vérifier qu'une valeur ne correspond pas au modèle indiqué).

Paramètres

Un paramètre spécifie dans l'expression un espace réservé associé à une valeur littérale renseignée au moment de l'exécution en affectant une valeur au tableau associatif SQLStatement.parameters. Les paramètres gèrent trois formes :

?	Un point d'interrogation indique un paramètre indexé. Des valeurs d'index numériques (de base zéro) sont affectées aux paramètres en fonction de leur ordre d'apparition dans l'instruction.
:AAAA	Un caractère deux-points suivi d'un nom d'identifiant conserve un espace au paramètre nommé AAAA. Les paramètres nommés sont également numérotés en fonction de leur ordre d'apparition dans l'instruction SQL. Pour éviter toute confusion, il est préférable d'éviter de combiner les paramètres numérotés et nommés.

@AAAA Le symbole arrobas est l'équivalent du caractère deux-points.

Fonctions SQL non prises en charge

La liste ci-dessous recense les éléments SQL standard qui ne sont pas pris en charge dans Adobe AIR :

Contraintes FOREIGN KEY Les contraintes FOREIGN KEY sont analysées, mais ne sont pas imposées.

Déclencheurs Les déclencheurs FOR EACH STATEMENT ne sont pas pris en charge (tous les déclencheurs doivent correspondre à FOR EACH ROW). Les déclencheurs INSTEAD OF ne sont pas pris en charge dans les tables (ils ne sont autorisés que dans les vues). Les déclencheurs récursifs (autrement dit, qui se déclenchent eux-mêmes) ne sont pas pris en charge.

ALTER TABLE Seules les variantes RENAME TABLE et ADD COLUMN de la commande ALTER TABLE sont prises en charge. Les autres types d'opérations ALTER TABLE, tels que DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT, etc. sont ignorés.

Transactions imbriquées Une seule transaction active est autorisée.

RIGHT OUTER JOIN et FULL OUTER JOIN RIGHT OUTER JOIN et FULL OUTER JOIN ne sont pas pris en charge.

VIEW actualisable Une vue est disponible en lecture seule. Il est impossible d'exécuter une instruction DELETE, INSERT ou UPDATE sur une vue. Un déclencheur INSTEAD OF exécuté lors d'une tentative d'instruction DELETE, INSERT ou UPDATE sur une vue est pris en charge et permet de mettre à jour les tables de soutien dans le corps du déclencheur.

GRANT et REVOKE Une base de données est un fichier de disque ordinaire ; les seules autorisations d'accès à appliquer sont les autorisations d'accès au fichier standard du système d'exploitation sous-jacent. Les commandes GRANT et REVOKE fréquemment détectées sur les systèmes SGBDR client/serveur ne sont pas implémentées.

Les éléments SQL et fonctionnalités SQLite suivants sont pris en charge dans certaines implémentations SQLite, mais ne le sont pas dans Adobe AIR. La plupart de ces fonctionnalités sont disponibles par le biais des méthodes de la classe SQLConnection :

Eléments SQL liés aux transactions (BEGIN, END, COMMIT, ROLLBACK) Cette fonctionnalité est disponible par le biais des méthodes liées aux transactions de la classe SQLConnection, SQLConnection.begin(), SQLConnection.commit() et SQLConnection.rollback().

ANALYZE Cette fonctionnalité est disponible par le biais de la méthode SQLConnection.analyze().

ATTACH Cette fonctionnalité est disponible par le biais de la méthode SQLConnection.attach().

COPY Cette instruction n'est pas prise en charge.

CREATE VIRTUAL TABLE Cette instruction n'est pas prise en charge.

DETACH Cette fonctionnalité est disponible par le biais de la méthode SQLConnection.detach().

PRAGMA Cette instruction n'est pas prise en charge.

VACUUM Cette fonctionnalité est disponible par le biais de la méthode SQLConnection.compact().

L'accès aux tables système n'est pas disponible. Les tables système, y compris sqlite_master et d'autres tables dotées du préfixe « sqlite_ » ne sont pas disponibles dans les instructions SQL. Le moteur d'exécution comprend une API de schéma qui permet d'accéder aux données du schéma par le biais d'une solution orientée objets. Pour plus d'informations, voir la méthode SQLConnection.loadSchema().

Fonctions d'expression régulière (MATCH() et REGEX()) Ces fonctions ne sont pas disponibles dans les instructions SQL.

La fonctionnalité suivante varie selon les nombreuses implémentations de SQLite et Adobe AIR :

Paramètres d'instruction indexés Dans de nombreuses implémentations, les paramètres d'instruction indexés sont basés sur un. Toutefois, dans Adobe AIR, les paramètres d'instruction indexés sont basés sur zéro (en d'autres termes, le premier paramètre est doté de l'index 0, le deuxième de l'index 1, etc.).

Définitions de colonne INTEGER PRIMARY KEY Dans de nombreuses implémentations, seules les colonnes définies exactement en tant qu'INTEGER PRIMARY KEY servent de colonne de clé primaire réelle de table. Dans ces implémentations, l'utilisation d'un type de données distinct, généralement synonyme d'INTEGER (tel qu'int), n'implique pas l'utilisation de la colonne comme clé primaire interne. Toutefois, dans Adobe AIR, le type de données int (ainsi que d'autres synonymes d'INTEGER) est considéré comme étant exactement équivalent à INTEGER. Par conséquent, une colonne définie en tant qu'INTEGER PRIMARY KEY sert de clé primaire interne de table. Pour plus d'informations, voir les sections CREATE TABLE et Affinité de colonne.

Autres fonctions SQL

Les types d'affinité de colonne suivants ne sont pas pris en charge par défaut dans SQLite, mais le sont dans Adobe AIR (notez qu'à l'instar de tous les mots-clés dans SQL, ces noms de types de données ne respectent pas la casse) :

Boolean correspond à la classe Boolean.

Date correspond à la classe Date.

int correspond à la classe int (équivaut à l'affinité de colonne INTEGER).

Number correspond à la classe Number (équivaut à l'affinité de colonne REAL).

Object correspond à la classe Object ou à toute sous-classe pouvant être sérialisée ou désérialisée par le biais d'AMF3. (Inclut la plupart des classes, y compris les classes personnalisées, mais exclut certaines classes telles que les objets d'affichage et les objets qui incluent des objets d'affichage en tant que propriétés.)

String correspond à la classe String (équivaut à l'affinité de colonne TEXT).

XML correspond à la classe XML d'ActionScript (E4X).

XMLList correspond à la classe XMLList d'ActionScript (E4X).

Les valeurs littérales suivantes ne sont pas prises en charge par défaut dans SQLite, mais le sont dans Adobe AIR :

true permet de représenter la valeur booléenne littérale true, avec les colonnes BOOLEAN.

false permet de représenter la valeur booléenne littérale false, avec les colonnes BOOLEAN.

Prise en charge des types de données

Contrairement à la plupart des bases de données SQL, le moteur de la base de données SQL d'Adobe AIR ne requiert pas ou n'impose pas que les colonnes des tables contiennent des valeurs d'un type donné. Le moteur d'exécution utilise plutôt deux concepts, les classes de stockage et l'affinité de colonne, pour contrôler les types de données. Cette section décrit les classes de stockage et l'affinité de colonne, ainsi que la résolution des différences de types de données selon les cas :

- « Classes de stockage » à la page 1178
- « Affinité de colonne » à la page 1178
- « Types de données et opérateurs de comparaison » à la page 1181
- « Types de données et opérateurs mathématiques » à la page 1181

- « Types de données et tri » à la page 1182
- « Types de données et regroupement » à la page 1182
- « Types de données et instructions SELECT composées » à la page 1182

Classes de stockage

Les classes de stockage représentent les types de données réels utilisés pour stocker des valeurs dans une base de données. La base de données fait appel aux classes de stockage suivantes :

NULL La valeur est une valeur NULL.

INTEGER La valeur est un nombre entier signé.

REAL La valeur est une valeur à virgule flottante.

TEXT La valeur est une chaîne texte (de 256 Mo au plus).

BLOB La valeur est un BLOB (Binary Large Object), c'est-à-dire des données binaires brutes (de 256 Mo au plus).

Toutes les valeurs, fournies à la base de données sous forme de valeurs littérales intégrées à une instruction SQL ou de valeurs liées à l'aide de paramètres à une instruction SQL préparée, se voient affecter une classe de stockage avant l'exécution de l'instruction SQL.

Les valeurs littérales qui font partie d'une instruction SQL se voient affecter la classe de stockage TEXT si elles sont placées entre guillemets simples ou doubles, INTEGER si la valeur littérale est un nombre sans guillemet et sans décimale ou exposant, REAL si la valeur littérale est un nombre sans guillemet avec décimale ou exposant et NULL si la valeur est NULL. Les valeurs littérales avec classe de stockage BLOB sont spécifiées par la notation X'ABCD'. Pour plus d'informations, voir la section consacrée aux valeurs littérales dans les expressions.

Les valeurs fournies en tant que paramètres via le tableau associatif SQLStatement.parameters se voient affecter la classe de stockage la plus proche de la liaison native du type de données. Par exemple, les valeurs int sont liées en tant que classe de stockage INTEGER, les valeurs Number obtiennent la classe de stockage REAL, les valeurs String obtiennent la classe de stockage TEXT et les objets ByteArray obtiennent la classe de stockage BLOB.

Affinité de colonne

L'*affinité* d'une colonne correspond au type recommandé des données stockées dans cette colonne. Lorsqu'une valeur est stockée dans une colonne (par une instruction INSERT ou UPDATE), le moteur d'exécution tente de convertir le type de données de cette valeur en l'affinité spécifiée. Par exemple, si une valeur Date (occurrence de Date ActionScript ou JavaScript) est insérée dans une colonne dont l'affinité est TEXT, elle est convertie en représentation String (ce qui équivaut à appeler la méthode toString() de l'objet) avant d'être stockée dans la base de données. Si la valeur ne peut pas être convertie en l'affinité spécifiée, une erreur est renvoyée et l'opération n'est pas exécutée. Lorsqu'une valeur est récupérée dans la base de données par une instruction SELECT, elle est renvoyée sous forme d'occurrence de la classe correspondant à l'affinité, qu'elle ait été ou non convertie à partir d'un autre type de données lors de son stockage.

Si une colonne accepte les valeurs NULL, la valeur null ActionScript ou JavaScript peut servir de valeur de paramètre pour stocker NULL dans la colonne. Lorsqu'une valeur de classe de stockage NULL est récupérée dans une instruction SELECT, elle est toujours renvoyée en tant que valeur null ActionScript ou JavaScript, quelle que soit l'affinité de la colonne. Lorsqu'une colonne accepte les valeurs NULL, vérifiez toujours les valeurs récupérées dans cette colonne pour déterminer si elles sont null avant de tenter d'attribuer aux valeurs un type qui ne peut pas être null (Number ou Boolean, par exemple).

L'une des affinités de type suivantes est affectée à chaque colonne de la base de données :

- TEXT (ou String)

- NUMERIC
- INTEGER (ou int)
- REAL (ou Number)
- Boolean
- Date
- XML
- XMLLIST
- Object
- NONE

TEXT (ou String)

Une colonne d'affinité TEXT ou String stocke toutes les données à l'aide des classes de stockage NULL, TEXT ou BLOB. Si des données numériques sont insérées dans une colonne dont l'affinité est TEXT, elles sont converties au format texte avant d'être stockées.

NUMERIC

Une colonne d'affinité NUMERIC stocke des valeurs à l'aide des classes de stockage NULL, REAL ou INTEGER. Lorsque des données texte sont insérées dans une colonne NUMERIC, le système tente de les convertir en nombres entiers ou réels avant de les stocker. Si la conversion aboutit, la valeur est stockée à l'aide de la classe de stockage INTEGER ou REAL (la valeur « 10,05 » est par exemple convertie en classe de stockage REAL avant d'être stockée). S'il est impossible d'effectuer la conversion, il se produit une erreur. Aucune tentative de conversion d'une valeur NULL est exécutée. Une valeur récupérée dans une colonne NUMERIC est renvoyée sous forme d'occurrence du type numérique le plus spécifique adapté à la valeur. En d'autres termes, si la valeur est un nombre entier positif ou 0, elle est renvoyée sous forme d'occurrence d'uint. S'il s'agit d'un nombre entier négatif, elle est renvoyée sous forme d'occurrence d'int. Enfin, s'il s'agit d'un composant à virgule flottante (et non d'un nombre entier), elle est renvoyée sous forme d'occurrence de Number.

INTEGER (ou int)

Une colonne qui utilise l'affinité INTEGER se comporte comme une colonne dont l'affinité est NUMERIC, à une exception près. Si la valeur à stocker est une valeur réelle (par exemple une occurrence de Number) sans composant à virgule flottante, ou si la valeur est une valeur de texte pouvant être convertie en valeur réelle sans composant à virgule flottante, elle est convertie en nombre entier et stockée à l'aide de la classe de stockage INTEGER. En cas de tentative de stockage d'une valeur réelle avec un composant à virgule flottante, une erreur se produit.

REAL (ou Number)

Une colonne dont l'affinité est REAL ou NUMBER se comporte comme une colonne d'affinité NUMERIC, sauf qu'elle convertit les valeurs entières en représentation à virgule flottante. Les valeurs d'une colonne REAL sont toujours renvoyées à partir de la base de données sous forme d'occurrences de Number.

Boolean

Une colonne d'affinité BOOLEAN stocke des valeurs true ou false. Une colonne BOOLEAN gère une valeur correspondant à une occurrence de Boolean ActionScript ou JavaScript. Si le code tente de stocker une valeur String, une valeur String dont la longueur est supérieure à zéro est considérée comme true, une valeur String vide comme false. Si le code tente de stocker des données numériques, toute valeur autre que zéro est stockée en tant que true et 0 est stocké en tant que false. Lorsqu'une valeur Boolean est récupérée par une instruction SELECT, elle est renvoyée en tant qu'occurrence de Boolean. Les valeurs non NULL sont stockées à l'aide de la classe de stockage INTEGER (0 pour false et 1 pour true) et sont converties en objet Boolean lors de la récupération des données.

Date

Une colonne d'affinité DATE stocke des valeurs de date et d'heure. Une colonne DATE est conçue pour accepter des valeurs qui sont des occurrences de Date ActionScript ou JavaScript. En cas de tentative de stockage d'une valeur String dans une colonne DATE, le moteur d'exécution tente de la convertir en date julienne. Si la conversion échoue, il se produit une erreur. Si le code tente de stocker une valeur Number, int ou uint, il ne se produit pas de tentative de validation des données et il est considéré comme acquis qu'il s'agit d'une valeur de date julienne valide. Une valeur DATE récupérée par une instruction SELECT est automatiquement convertie en occurrence de Date. Les valeurs DATE étant stockées en tant que valeurs de dates juliennes avec la classe de stockage REAL, les opérations de tri et de comparaison fonctionnent comme prévu.

XML ou XMLList

Une colonne qui utilise une affinité XML ou XMLLIST stocke des structures XML. Lorsque le code tente de stocker des données dans une colonne XML en utilisant un paramètre SQLStatement, le moteur d'exécution tente de convertir et de valider la valeur à l'aide de la fonction ActionScript XML() ou XMLList(). Si la valeur ne peut pas être convertie en code XML valide, une erreur se produit. Si la tentative de stockage des données utilise une valeur de texte SQL littérale (INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)', par exemple), la valeur n'est ni analysée, ni validée, car son format est considéré comme correct. Si une valeur non valide est stockée, elle est renvoyée en tant qu'objet XML vide lorsqu'elle est récupérée. Les données XML et XMLLIST sont stockées à l'aide de la classe de stockage TEXT ou NULL.

Object

Une colonne d'affinité Object stocke des objets complexes ActionScript ou JavaScript, y compris des occurrences de la classe Object, des occurrences des sous-classes Object telles que les occurrences d'Array, voire des occurrences des classes personnalisées. Les données de la colonne Object sont sérialisées au format AMF3 et stockées avec la classe de stockage BLOB. Lorsqu'une valeur est récupérée, elle est désérialisée du format AMF3 et renvoyée sous forme d'occurrence de leur classe d'origine. Notez que certaines classes ActionScript, en particulier les objets d'affichage, ne peuvent pas être désérialisées en tant qu'occurrences de leur type de données d'origine. Avant de stocker une occurrence de classe personnalisée, vous devez enregistrer un alias pour la classe avec la méthode flash.net.registerClassAlias() (ou dans Flex, en ajoutant des métadonnées [RemoteObject] à la déclaration de la classe). Avant de récupérer les données, vous devez également enregistrer le même alias pour la classe. Toutes les données qui ne peuvent pas être désérialisées correctement, que le problème soit inhérent à la classe ou dû au fait qu'un alias de classe est introuvable ou ne correspond pas à la valeur requise, sont renvoyées sous forme d'objet anonyme (une occurrence de la classe Object) dont les propriétés et les valeurs correspondent à l'occurrence d'origine telle qu'elle a été stockée.

NONE

Une colonne d'affinité NONE ne privilégie aucune classe de stockage. Elle ne tente pas de convertir les données avant de les insérer.

Identification de l'affinité

L'affinité de type d'une colonne est déterminée par le type déclaré de la colonne dans l'instruction CREATE TABLE. Lors de la détermination du type, les règles suivantes (non sensibles à la casse) sont appliquées :

- Si le type de données de la colonne contient l'une des chaînes « CHAR », « CLOB », « STRI » ou « TEXT », cette colonne est alors d'affinité TEXT/STRING. Notez que le type VARCHAR contient la chaîne « CHAR » et possède donc l'affinité TEXT.
- Si le type de données de la colonne contient la chaîne « BLOB » ou si aucun type de données n'est spécifié, l'affinité de la colonne est NONE.
- Si le type de données de la colonne contient la chaîne « XMLL », l'affinité de la colonne est XMLList.

- Si le type de données correspond à la chaîne « XML », l'affinité de la colonne est XML.
- Si le type de données contient la chaîne « OBJE », l'affinité de la colonne est Object.
- Si le type de données contient la chaîne « BOOL », l'affinité de la colonne est Boolean.
- Si le type de données contient la chaîne « DATE», l'affinité de la colonne est Date.
- Si le type de données contient la chaîne « INT» (y compris « UINT »), l'affinité de la colonne est INTEGER/int.
- Si le type de données d'une colonne contient la chaîne « REAL », « NUMB », « FLOA » ou « DOUB », l'affinité de la colonne est REAL/Number.
- Dans le cas contraire, l'affinité est NUMERIC.
- Si une table est créée à l'aide d'une instruction CREATE TABLE t AS SELECT..., aucun type de données n'est spécifié pour les colonnes et l'affinité NONE leur est affectée.

Types de données et opérateurs de comparaison

Les opérateurs de comparaison binaires suivants, =, <, <=, >= et != sont pris en charge, ainsi qu'une opération permettant de tester l'appartenance, IN, et l'opérateur de comparaison ternaire BETWEEN. Pour plus d'informations sur ces opérateurs, voir Opérateurs.

Le résultat d'une comparaison dépend des classes de stockage des deux valeurs comparées. Lors de la comparaison de deux valeurs, les règles suivantes sont appliquées :

- Une valeur dont la classe de stockage est NULL est considérée inférieure à toute autre valeur (y compris à une autre valeur dont la classe de stockage est NULL).
- Une valeur INTEGER ou REAL est inférieure à toute valeur TEXT ou BLOB. Lorsqu'une valeur INTEGER ou REAL est comparée à une autre valeur INTEGER ou REAL, une comparaison numérique est effectuée.
- Une valeur TEXT est inférieure à une valeur BLOB. Lorsque deux valeurs TEXT sont comparées, une comparaison binaire est effectuée.
- Lorsque deux valeurs BLOB sont comparées, le résultat est toujours déterminé par une comparaison binaire.

L'opérateur ternaire BETWEEN est toujours remanié en tant qu'expression binaire équivalente. Par exemple, a BETWEEN b AND c est transformé en a >= b AND a <= c, même si cela signifie que des affinités différentes sont appliquées à a dans chacune des comparaisons requises pour évaluer l'expression.

Les expressions de type a IN (SELECT b ....) sont gérées selon les trois règles énumérées précédemment pour les comparaisons binaires, c'est-à-dire de la même façon que a = b. Par exemple, si b est une valeur de colonnes et a une expression, l'affinité de b est appliquée à a avant toute comparaison. L'expression a IN (x, y, z) est transformée en a = +x OR a = +y OR a = +z. Les valeurs situées à droite de l'opérateur IN (soit les valeurs x, y et z dans cet exemple) sont considérées comme des expressions, même s'il s'agit de valeurs de colonne. Si la valeur située à gauche de l'opérateur IN est une colonne, l'affinité de cette dernière est utilisée. Si la valeur est une expression, il ne se produit pas de conversion.

L'utilisation de la clause COLLATE peut également affecter la manière dont les comparaisons sont effectuées. Pour plus d'informations, voir COLLATE.

Types de données et opérateurs mathématiques

Pour chaque opérateur mathématique pris en charge (*, /, %, + et -), l'affinité numérique est appliquée à chaque opérande avant d'évaluer l'expression. S'il est impossible de convertir un opérande en classe de stockage NUMERIC, l'évaluation de l'expression donne le résultat NULL.

Si l'opérateur de concaténation || est utilisé, chaque opérande est converti en classe de stockage TEXT avant que l'expression ne soit évaluée. S'il est impossible de convertir un opérande en classe de stockage TEXT, le résultat de l'expression est NULL. Il peut s'avérer impossible de convertir la valeur dans deux cas de figure : soit la valeur de l'opérande est NULL, soit il s'agit d'un objet BLOB contenant une classe de stockage dont le type est autre que TEXT.

Types de données et tri

Lorsque les valeurs sont triées par une clause ORDER BY, les valeurs associées à la classe de stockage NULL sont prioritaires. Elles sont suivies des valeurs INTEGER et REAL classées par ordre numérique, suivies des valeurs TEXT classées par ordre binaire ou selon le classement spécifié (BINARY ou NOCASE). Les valeurs BLOB arrivent en dernière position, classées par ordre binaire. Il ne se produit pas de conversion des classes de stockage avant le tri.

Types de données et regroupement

Lors d'un regroupement des valeurs à l'aide de la clause GROUP BY, les valeurs associées à des classes de stockage différentes sont considérées comme distinctes. Les valeurs INTEGER et REAL font exception à la règle, car elles sont considérées comme égales si elles sont équivalentes d'un point de vue numérique. Aucune affinité n'est appliquée aux valeurs suite à l'exécution d'une clause GROUP BY.

Types de données et instructions SELECT composées

Les opérateurs SELECT composés (UNION, INTERSECT et EXCEPT) effectuent des comparaisons implicites entre les valeurs. Avant l'exécution de ces comparaisons, une affinité est susceptible d'être appliquée à chaque valeur. La même infinité est appliquée le cas échéant à toutes les valeurs pouvant être renvoyées dans une seule colonne du jeu de résultats SELECT composé. L'affinité appliquée correspond à l'affinité de la colonne renvoyée par la première instruction SELECT de composant dont une valeur de colonne (et non un autre type d'expression) occupe cette position. Si, pour une colonne SELECT composée donnée, aucune instruction SELECT de composant ne renvoie de valeur de colonne, aucune affinité n'est appliquée aux valeurs de cette colonne avant leur comparaison.

Chapitre 67 : ID, arguments et messages d'erreur SQL détaillés

La classe SQLError représente les diverses erreurs susceptibles de se produire lors de l'utilisation d'une base de données SQL locale Adobe AIR. Pour toute exception donnée, l'occurrence de SQLError possède une propriété `details` qui contient un message d'erreur en anglais. Chaque message d'erreur est en outre associé à un identifiant unique disponible dans la propriété `detailID` de l'objet SQLError. Grâce à la propriété `detailID`, une application peut identifier le message d'erreur spécifique de la propriété details. L'application peut alors fournir un autre texte à l'utilisateur, dans sa langue. Les valeurs des arguments du tableau `detailArguments` peuvent être substituées à la position appropriée dans la chaîne du message d'erreur. Cette caractéristique s'avère utile pour les applications qui affichent directement le message d'erreur de la propriété `details` à l'intention des utilisateurs finaux dans une langue spécifique.

Le tableau ci-dessous contient la liste des valeurs `detailID` et le texte du message d'erreur associé en anglais. Le texte d'espace réservé dans les messages indique l'emplacement où sont substituées les valeurs de `detailArguments` par le moteur d'exécution. Cette liste peut servir de source de localisation des messages d'erreur susceptibles d'être générés lors d'opérations de bases de données SQL.

SQLError detailID	Message d'erreur détaillé et paramètres en anglais
1001	Connection closed. (Connexion fermée)
1102	Database must be open to perform this operation. (Pour effectuer cette opération, la base de données doit être ouverte.)
1003	%s [,|and %s] parameter name(s) found in parameters property but not in the SQL specified. (Les noms de paramètres %s [,| et %s] ont été détectés dans la propriété parameters, mais non dans l'instruction SQL spécifiée.)
1004	Mismatch in parameter count. Found %d in SQL specified and %d value(s) set in parameters property. Expecting values for %s [,|and %s]. (Le nombre de paramètres ne correspond pas. %d détecté dans l'instruction SQL spécifiée et valeur(s) %d définie(s) dans la propriété parameters. Attente de valeurs pour %s [,| et %s].)
1005	Auto compact could not be turned on. (Le compactage automatique n'a pas pu être activé.)
1006	The pageSize value could not be set. (La valeur pageSize n'a pas pu être définie.)
1007	The schema object with name '%s' of type '%s' in database '%s' was not found. (L'objet schema nommé '%s' de type '%s' est introuvable dans la base de données '%s'.)
1008	The schema object with name '%s' in database '%s' was not found. (L'objet schema nommé '%s' est introuvable dans la base de données '%s'.)
1009	No schema objects with type '%s' in database '%s' were found.(Aucun objet schema de type '%s' n'a été détecté dans la base de données '%s'.)
1010	No schema objects in database '%s' were found. (Aucun objet schema n'a été détecté dans la base de données '%s'.)
2001	Parser stack overflow. (Dépassement de capacité de la pile de l'analyseur.)
2002	Too many arguments on function '%s' (Trop d'arguments dans la fonction '%s'.)
2003	near '%s': syntax error (à proximité de '%s' : erreur de syntaxe)
2004	there is already another table or index with this name: '%s' (une autre table ou un autre index porte déjà ce nom : '%s')
2005	PRAGMA is not allowed in SQL. (PRAGMA n'est pas autorisé dans SQL.)
2006	Not a writable directory. (Il est impossible d'écrire dans ce répertoire.)
2007	Unknown or unsupported join type: '%s %s %s' (Type de jointure inconnu ou non pris en charge : '%s %s %s')
2008	RIGHT and FULL OUTER JOINs are not currently supported. (Les jointures RIGHT et FULL OUTER ne sont pas prises en charge actuellement.)
2009	A NATURAL join may not have an ON or USING clause. (Il est possible qu'une jointure NATURAL n'ait pas de clause ON ou USING.)

2010	Cannot have both ON and USING clauses in the same join (Une même jointure ne peut pas contenir à la fois les clauses ON et USING.)
2011	Cannot join using column '%s' - column not present in both tables. (Jointure impossible avec la colonne '%s' - la colonne est absente des deux tables.)
2012	Only a single result allowed for a SELECT that is part of an expression. (Un seul résultat est autorisé pour la clause SELECT faisant partie d'une expression.)
2013	No such table: '[%s.]%s' (Il n'existe pas de table : '[%s.]%s')
2014	No tables specified. (Aucune table spécifiée.)
2015	Too many columns in result set|too many columns on '%s'. (Trop de colonnes dans le jeu de résultats|trop de colonnes dans '%s'.)
2016	%s ORDER|GROUP BY term out of range - should be between 1 and %d (Le terme %s ORDER|GROUP BY est hors limites ; il doit être compris entre 1 et %d.)
2017	Too many terms in ORDER BY clause. (La clause ORDER BY contient trop de termes.)
2018	%s ORDER BY term out of range - should be between 1 and %d. (Le terme %s ORDER BY est hors limites ; il doit être compris entre 1 et %d.)
2019	%r ORDER BY term does not match any column in the result set. (Le terme %r ORDER BY ne correspond à aucune colonne du jeu de résultats.)
2020	ORDER BY clause should come after '%s' not before. (La clause ORDER BY doit être insérée après '%s' et non avant.)
2021	LIMIT clause should come after '%s' not before. (La clause LIMIT doit être insérée après '%s' et non avant.)
2022	SELECTs to the left and right of '%s' do not have the same number of result columns. (Les clauses SELECT placées à gauche et à droite de '%s' n'ont pas le même nombre de colonnes de résultats.)
2023	A GROUP BY clause is required before HAVING. (Une clause GROUP BY est requise avant HAVING.)
2024	Aggregate functions are not allowed in the GROUP BY clause. (Les fonctions d'agrégation ne sont pas autorisées dans la clause GROUP BY.)
2025	DISTINCT in aggregate must be followed by an expression. (DISTINCT doit être suivi d'une expression dans une fonction d'agrégation.)
2026	Too many terms in compound SELECT. (La clause SELECT composée contient trop de termes.)
2027	Too many terms in ORDER|GROUP BY clause (La clause ORDER|GROUP BY contient trop de termes.)
2028	Temporary trigger may not have qualified name (Le déclencheur temporaire n'a peut-être pas de nom complet)
2030	Trigger '%s' already exists (Le déclencheur '%s' existe déjà)
2032	Cannot create BEFORE|AFTER trigger on view: '%s'. (Impossible de créer un déclencheur BEFORE|AFTER sur la vue : '%s'.)
2033	Cannot create INSTEAD OF trigger on table: '%s'. (Impossible de créer un déclencheur INSTEAD OF sur la table : '%s'.)
2034	No such trigger: '%s' (Ce déclencheur n'existe pas : '%s')
2035	Recursive triggers not supported ('%s'). (Les déclencheurs récursifs ne sont pas pris en charge ('%s').)
2036	No such column: %s[.%s[.%s]] (Cette colonne n'existe pas : %s[.%s[.%s]])
2037	VACUUM is not allowed from SQL. (VACUUM n'est pas autorisé depuis une instruction SQL.)
2043	Table '%s': indexing function returned an invalid plan. (Table '%s' : une fonction d'indexation a renvoyé un plan non valide.)
2044	At most %d tables in a join. (Une jointure ne doit pas comporter plus de %d tables.)
2046	Cannot add a PRIMARY KEY column. (Impossible d'ajouter une colonne PRIMARY KEY.)
2047	Cannot add a UNIQUE column. (Impossible d'ajouter une colonne UNIQUE.)
2048	Cannot add a NOT NULL column with default value NULL. (Impossible d'ajouter une colonne NOT NULL avec la valeur par défaut NULL.)
2049	Cannot add a column with non-constant default. (Impossible d'ajouter une colonne avec une valeur par défaut non constante.)
2050	Cannot add a column to a view. (Impossible d'ajouter une colonne à une vue.)
2051	ANALYZE is not allowed in SQL. (ANALYZE n'est pas autorisé dans SQL.)
2052	Invalid name: '%s' (Nom non valide : '%s')

2053	ATTACH is not allowed from SQL. (ATTACH n'est pas autorisé depuis une instruction SQL.)
2054	%s '%s' cannot reference objects in database '%s' (%s '%s' ne peut pas référencer des objets dans la base de données '%s')
2055	Access to '[%s.]%s.%s' is prohibited. (L'accès à '[%s.]%s.%s' est interdit.)
2056	Not authorized. (Interdit.)
2058	No such view: '[%s.]%s' (Il n'existe pas de vue : '[%s.]%s')
2060	Temporary table name must be unqualified. (Le nom de la table temporaire doit être incomplet.)
2061	Table '%s' already exists. (La table '%s' existe déjà.)
2062	There is already an index named: '%s' (Il existe déjà un index nommé '%s'.)
2064	Duplicate column name: '%s' (Nom de colonne en double : '%s')
2065	Table '%s' has more than one primary key. (La table '%s' possède plusieurs clés primaires.)
2066	AUTOINCREMENT is only allowed on an INTEGER PRIMARY KEY (AUTOINCREMENT n'est autorisé que sur un élément INTEGER PRIMARY KEY)
2067	No such collation sequence: '%s' (La séquence de classement n'existe pas : '%s')
2068	Parameters are not allowed in views. (Les paramètres sont interdits dans les vues.)
2069	View '%s' is circularly defined. (La vue '%s' est définie circulairement.)
2070	Table '%s' may not be dropped. (Il est interdit de supprimer la table '%s'.)
2071	Use DROP VIEW to delete view '%s' (Utilisez DROP VIEW pour supprimer la vue '%s'.)
2072	Use DROP TABLE to delete table '%s' (Utilisez DROP TABLE pour supprimer la table '%s')
2073	Foreign key on '%s' should reference only one column of table '%s' (La clé étrangère associée à '%s' doit référencer une seule colonne de la table '%s')
2074	Number of columns in foreign key does not match the number of columns in the referenced table. (Le nombre de colonnes de la clé étrangère ne correspond pas au nombre de colonnes de la table référencée.)
2075	Unknown column '%s' in foreign key definition. (Colonne inconnue '%s' dans la définition de la clé étrangère.)
2076	Table '%s' may not be indexed. (Il est interdit d'indexer la table '%s'.)
2077	Views may not be indexed. (Il est interdit d'indexer les vues.)
2080	Conflicting ON CONFLICT clauses specified. (Les clauses ON CONFLICT spécifiées sont sources de conflit.)
2081	No such index: '%s' (Cet index n'existe pas : '%s')
2082	Index associated with UNIQUE or PRIMARY KEY constraint cannot be dropped. (L'index associé à la contrainte UNIQUE ou PRIMARY KEY ne peut pas être abandonné.)
2083	BEGIN is not allowed in SQL. (BEGIN n'est pas autorisé dans SQL.)
2084	COMMIT is not allowed in SQL. (COMMIT n'est pas autorisé dans SQL.)
2085	ROLLBACK is not allowed in SQL. (ROLLBACK n'est pas autorisé dans SQL.)
2086	Unable to open a temporary database file for storing temporary tables. (Impossible d'ouvrir un fichier de base de données temporaire pour stocker des tables temporaires.)
2087	Unable to identify the object to be reindexed. (Impossible d'identifier l'objet à réindexer.)
2088	Table '%s' may not be modified. (Il est interdit de modifier la table '%s'.)
2089	Cannot modify '%s' because it is a view. (Impossible de modifier '%s' car il s'agit d'une vue.)
2090	Variable number must be between ?0 and ?%d< (Le nombre de variables doit être compris entre ?0 et ?%d<)
2092	Misuse of aliased aggregate '%s' (Utilisation incorrecte de l'agrégat crénelé '%s')
2093	Ambiguous column name: '[%s.[%s.]]%s' (Nom de colonne ambigu : '[%s.[%s.]]%s')
2094	No such function: '%s' (Cette fonction n'existe pas : '%s')
2095	Wrong number of arguments to function '%s' (Nombre incorrect d'arguments pour la fonction '%s')
2096	Subqueries prohibited in CHECK constraints. (Les sous-requêtes sont interdites dans les contraintes CHECK.)
2097	Parameters prohibited in CHECK constraints. (Les paramètres sont interdits dans les contraintes CHECK.)

2098	Expression tree is too large (maximum depth %d) (L'arborescence de l'expression est trop importante (la profondeur maximale est de %d))
2099	RAISE() may only be used within a trigger-program (RAISE() ne peut être utilisé que dans un programme déclencheur)
2100	Table '%s' has %d columns but %d values were supplied (La table '%s' possède '%s' colonnes, mais %d valeurs ont été fournies.)
2101	Database schema is locked: '%s' (Le schéma de base de données est verrouillé : '%s')
2102	Statement too long. (L'instruction est trop longue.)
2103	Unable to delete/modify collation sequence due to active statements (Impossible de supprimer/modifier la séquence de classement en raison d'instructions actives)
2104	Too many attached databases - max %d (Trop de bases de données jointes ; maxi. %d)
2105	Cannot ATTACH database within transaction. (Impossible de joindre une base de données à la transaction via ATTACH.)
2106	Database '%s' is already in use. (La base de données '%s' est déjà utilisée.)
2108	Attached databases must use the same text encoding as main database. (Les bases de données jointes doivent utiliser le même codage de texte que la base de données principale.)
2200	Out of memory. (Mémoire insuffisante.)
2201	Unable to open database. (Impossible d'ouvrir la base de données.)
2202	Cannot DETACH database within transaction. (Impossible de détacher une base de données de la transaction via DETACH.)
2203	Cannot detach database: '%s' (Impossible de détacher la base de données : '%s')
2204	Database '%s' is locked. (La base de données '%s' est verrouillée.)
2205	Unable to acquire a read lock on the database. (Impossible d'obtenir un verrouillage en lecture sur la base de données.)
2206	[column|columns] '%s'[,'%s'] are not [unique|is] not unique. (La/les [colonne|colonnes] '%s'[,'%s'] n'est pas [unique|ne sont] pas uniques.)
2207	Malformed database schema. (Schéma de base de données incorrectement formé.)
2208	Unsupported file format. (Format de fichier non pris en charge.)
2209	Unrecognized token: '%s' (Jeton non reconnu : '%s')
2300	Could not convert text value to numeric value. (Impossible de convertir la valeur de texte en valeur numérique.)
2301	Could not convert string value to date. (Impossible de convertir la valeur de chaîne en date.)
2302	Could not convert floating point value to integer without loss of data. (Impossible de convertir la valeur en virgule flottante en nombre entier sans perte de données.)
2303	Cannot rollback transaction - SQL statements in progress. (Impossible d'annuler la transaction - instructions SQL en cours.)
2304	Cannot commit transaction - SQL statements in progress. (Impossible d'envoyer la transaction - instructions SQL en cours.)
2305	Database table is locked: '%s' (La table de base de données est verrouillée : '%s')
2306	Read-only table. (Table en lecture seule.)
2307	String or blob too big. (Chaîne ou blob trop long.)
2309	Cannot open indexed column for writing. (Impossible d'ouvrir une colonne indexée en écriture.)
2400	Cannot open value of type %s. (Impossible d'ouvrir une valeur de type %s.)
2401	No such rowid: %s< (Cet ID de ligne n'existe pas : %s<)
2402	Object name reserved for internal use: '%s' (Nom d'objet réservé à un usage interne : '%s')
2403	View '%s' may not be altered. (Impossible de modifier la vue '%s'.)
2404	Default value of column '%s' is not constant. (La valeur par défaut de la colonne '%s' n'est pas constante.)
2405	Not authorized to use function '%s' (L'utilisation de la fonction '%s' n'est pas autorisée)
2406	Misuse of aggregate function '%s' (Utilisation incorrecte de la fonction d'agrégation '%s')
2407	Misuse of aggregate '%s' (Utilisation incorrecte de l'agrégation '%s')
2408	No such database: '%s' (Cette base de données n'existe pas : '%s')
2409	Table '%s' has no column named '%s' (La table '%s' ne contient pas de colonne nommée '%s')

2501	No such module: '%s' (Ce module n'existe pas : '%s')
2508	No such savepoint: '%s' (Ce point de sauvegarde n'existe pas : '%s')
2510	Cannot rollback - no transaction is active. (Annulation impossible ; aucune transaction n'est active.)
2511	Cannot commit - no transaction is active. (Confirmation impossible ; aucune transaction n'est active.)

Chapitre 68 : AGAL (Adobe Graphics Assembly Language)

AGAL (Adobe Graphics Assembly Language) est un langage de shaders permettant de définir des programmes de rendu de sommets et de fragments. Les programmes AGAL doivent être téléchargés dans le contexte de rendu au format de pseudo-code binaire décrit dans ce document.

Pseudo-code binaire AGAL

Le pseudo-code AGAL doit utiliser le format Endian.LITTLE_ENDIAN.

En-tête du pseudo-code

Le pseudo-code AGAL doit commencer par un en-tête de 7 octets :

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

Décalage (octets)	Taille (octets)	Nom	Description
0	1	magic	doit être 0xa0
1	4	version	doit être 1
5	1	shader type ID	doit être 0xa1
6	1	shader type	0 pour un programme de sommets, 1 pour un programme de fragments

Jetons

L'en-tête est immédiatement suivi par un nombre quelconque de jetons. Chaque jeton est de 192 bits (24 octets) et a toujours le format suivant :

```
[opcode][destination][source1][source2 or sampler]
```

Tous les codes d'opération n'utilisent pas l'ensemble de ces champs. Les champs non utilisés doivent être définis sur 0.

Codes d'opération

Le champ [opcode] est de 32 bits et peut prendre l'une des valeurs suivantes :

Nom	Code d'op.	Opération	Description
mov	0x00	déplacer	déplacer les données à partir de source1 vers destination, au niveau du composant
add	0x01	ajouter	destination = source1 + source2, au niveau du composant
sub	0x02	soustraire	destination = source1 - source2, au niveau du composant
mul	0x03	multiplier	destination = source1 * source2, au niveau du composant

Nom	Code d'op.	Opération	Description
div	0x04	diviser	destination = source1 / source2, au niveau du composant
rcp	0x05	réciproque	destination = 1/source1, au niveau du composant
min	0x06	minimum	destination = minimum(source1,source2), au niveau du composant
max	0x07	maximum	destination = maximum(source1,source2), au niveau du composant
frc	0x08	fractionnaire	destination = source1 - (float)floor(source1), au niveau du composant
sqt	0x09	racine carrée	destination = sqrt(source1), au niveau du composant
rsq	0x0a	racine réciproque	destination = 1/sqrt(source1), au niveau du composant
pow	0x0b	puissance	destination = pow(source1,source2), au niveau du composant
log	0x0c	logarithme	destination = log_2(source1), au niveau du composant
exp	0x0d	exponentiel	destination = 2^source1, au niveau du composant
nrm	0x0e	normaliser	destination = normalize(source1), au niveau du composant (produit uniquement un résultat de composant 3, la destination doit être masquée sur .xyz ou inférieur)
sin	0x0f	sinus	destination = sin(source1), au niveau du composant
cos	0x10	cosinus	destination = cos(source1), au niveau du composant
crs	0x11	produit vectoriel	destination.x = source1.y * source2.z - source1.z * source2.y destination.y = source1.z * source2.x - source1.x * source2.z destination.z = source1.x * source2.y - source1.y * source2.x (produit uniquement un résultat de composant 3, la destination doit être masquée sur .xyz ou inférieur)
dp3	0x12	produit scalaire	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z
dp4	0x13	produit scalaire	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z + source1.w*source2.w
abs	0x14	absolu	destination = abs(source1), au niveau du composant
neg	0x15	inverser	destination = -source1, au niveau du composant
sat	0x16	saturer	destination = maximum(minimum(source1,1),0), au niveau du composant
m33	0x17	multiplier la matrice 3x3	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) (produit uniquement un résultat de composant 3, la destination doit être masquée sur .xyz ou inférieur)

Nom	Code d'op.	Opération	Description
m44	0x18	multiplier la matrice 4x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) destination.w = (source1.x * source2[3].x) + (source1.y * source2[3].y) + (source1.z * source2[3].z) + (source1.w * source2[3].w)
m34	0x19	multiplier la matrice 3x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) (produit uniquement un résultat de composant 3, la destination doit être masquée sur .xyz ou inférieur)
kil	0x27	tuer/supprimer (shader de fragments uniquement)	Si le composant source scalaire unique est inférieur à zéro, le fragment est supprimé et non dessiné dans le tampon d'images. (Le registre de destination doit être défini sur tous les 0)
tex	0x28	échantillon de texture (shader de fragments uniquement)	destination équivaut à charger depuis la texture source2 aux coordonnées source1. Dans ce cas, source2 doit être au format de l'échantillonneur.
sge	0x29	définir si supérieur ou égal à	destination = source1 >= source2 ? 1 : 0, au niveau du composant
slt	0x2a	définir si inférieur ou égal à	destination = source1 < source2 ? 1 : 0, au niveau du composant
seq	0x2c	définir si égal à	destination = source1 == source2 ? 1 : 0, au niveau du composant
sne	0x2d	définir si non égal à	destination = source1 != source2 ? 1 : 0, au niveau du composant

Format du champ Destination

Le champ [destination] est de 32 bits :

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = type de registre (4 bits)

M = masque d'écriture (4 bits)

N = numéro de registre (16 bits)

- = non défini, doit être 0

Format du champ Source

Le champ [source] est de 64 bits :

```
63.............................................................0
D------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

D = direct=0/indirect=1 pour direct Q et I sont ignorés, 1 bit

Q = sélection du composant du registre d'index (2 bits)

I = type de registre d'index (4 bits)

T = type de registre (4 bits)

S = réagencement des données en surface (8 bits, 2 bits par composant)

O = décalage indirect (8 bits)

N = numéro de registre (16 bits)

- = non défini, doit être 0

Format du champ Sampler

Le deuxième champ source correspondant au code d'opération tex doit être au format [sampler], qui est de 64 bits :

```
63.............................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = numéro de registre de l'échantillonneur (16 bits)

B = valeur de niveau de détail de la texture, entier signé, échelle par 8. La valeur à virgule flottante utilisée est b/8.0 (8 bits)

T = type de registre, doit être 5, Sampler (4 bits)

F = filtre (0=le plus proche,1=linéaire) (4 bits)

M = mipmap (0=désactivé,1=le plus proche, 2=linéaire)

W = enveloppe (0=verrouillage,1=répétition)

S = Bits d'indicateurs spéciaux (doivent être 0)

D = dimension (0=2D, 1=Cube)

Registres de programme

Les types de registres suivants sont définis : Utilisez la valeur répertoriée pour spécifier un type de registre dans les champs d'un jeton :

Nom	Valeur	Nombre par programme de fragments	Nombre par programme de sommets	Utilisation
Attribut	0	sans objet	8	Entrée de shader de sommets ; à lire à partir d'un tampon de sommets spécifié à l'aide de Context3D.setVertexBufferAt().
Constante	1	28	128	Entrée de shader ; à définir à l'aide de la famille de fonctions Context3D.setProgramConstants().
Temporaire	2	8	8	Registre temporaire pour le calcul, non accessible en dehors du programme.

Nom	Valeur	Nombre par programme de fragments	Nombre par programme de sommets	Utilisation
Sortie	3	1	1	Sortie de shader : dans un programme de sommets, la sortie correspond à la position de l'espace du clip ; dans un programme de fragments, la sortie est une couleur.
Variable	4	8	8	Transférez les données interpolées entre les shaders de sommets et de fragments. Les registres de variables du programme de sommets sont appliqués en tant qu'entrée au programme de fragments. Les valeurs sont interpolées en fonction de la distance par rapport aux sommets du triangle.
Echantillonneur	5	8	sans objet	Entrée de shader de fragments ; à lire à partir d'une texture spécifiée à l'aide de Context3D.setTextureAt().