ACTIONSCRIPT® 3.0-ontwikkelaarsgids

Juridische kennisgeving

Laatst bijgewerkt 14/10/2015

Inhoud

Hoofdstuk 24: Werken met geluid

Hoofdstuk 25: Werken met video

Hoofdstuk 26: Werken met camera's

Hoofdstuk 27: Digitaal rechtenbeheer gebruiken

Hoofdstuk 59: HTML en JavaScript programmeren in AIR

Hoofdstuk 60: Scripting van de AIR HTML-container

Hoofdstuk 61: Aan HTML gerelateerde gebeurtenissen in AIR afhandelen

Hoofdstuk 62: HTML-inhoud weergeven in mobiele toepassingen

Hoofdstuk 63: Workers voor gelijktijdige uitvoering gebruiken

Hoofdstuk 64: Beveiliging

Hoofdstuk 1: Werken met datums en tijden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Timing is misschien niet alles, maar het is doorgaans een essentiële factor in softwaretoepassingen. ActionScript 3.0 biedt krachtige methoden voor het beheren van kalenderdatums, -tijden en -tijdintervallen. Het grootste deel van de tijdfunctionaliteit wordt verzorgd door twee belangrijke klassen: de klasse Date en de nieuwe klasse Timer in het pakket flash.utils.

Datums en tijden worden veelvuldig gebruikt in ActionScript-programma's. Het kan bijvoorbeeld voorkomen dat u moet weten welke dag het vandaag is of moet bijhouden hoelang een gebruiker een bepaald scherm heeft geopend. In ActionScript kunt u de klasse Date gebruiken om één moment in de tijd te vertegenwoordigen. De klasse bevat tevens datum- en tijdgegevens. Een instantie Date bevat waarden voor de afzonderlijke datum- en tijdeenheden, inclusief het jaar, de maand, de datum, de dag van de week, het uur, minuten, seconden, milliseconden en de tijdzone. Voor meer geavanceerde toepassingen is in ActionScript de klasse Timer opgenomen. Deze kunt u gebruiken om bepaalde handelingen na een bepaalde vertraging of met regelmatige tussenpozen uit te voeren.

Meer Help-onderwerpen

Date

flash.utils.Timer

Kalenderdatums en -tijden beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 3.0 bevinden alle functies voor het beheren van kalenderdatums en -tijden zich in de klasse Date op hoofdniveau. De klasse Date bevat methoden en eigenschappen waarmee datums en tijden in UTC (Coordinated Universal Time) of in de lokale tijd kunnen worden verwerkt. UTC is net als GMT (Greenwich Mean Time) een standaarddefinitie voor tijd.

Date-objecten maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Date bevat een van de meest veelzijdige constructormethoden van alle kernklassen. U kunt deze op vier manieren aanroepen:

Eerste methode: als er geen parameters zijn opgegeven, retourneert de constructor `Date()` een object Date met de huidige datum en tijd voor uw tijdzone. Hier volgt een voorbeeld:

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

Tweede methode: als er een numerieke parameter is opgegeven, wordt deze waarde door de constructor `Date()` beschouwd als het aantal milliseconden sinds 1 januari 1970 en wordt er een overeenkomend object Date geretourneerd. De waarde die u doorgeeft, wordt beschouwd als het aantal milliseconden sinds 1 januari 1970, in UTC. In het object Date worden waarden echter in uw lokale tijdzone weergegeven, tenzij u de UTC-specifieke methoden gebruikt om ze op te halen en weer te geven. Als u een nieuw object Date maakt met één parameter voor milliseconden, moet u rekening met het tijdverschil tussen uw lokale tijd en UTC. Met de volgende instructies wordt een object Date op middernacht op 1 januari 1970 (UTC) ingesteld:

```
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);
```

Derde methode: u kunt meerdere numerieke parameters doorgeven aan de constructor `Date()`. Deze parameters worden respectievelijk als het jaar, de maand, de dag, het uur, de minuut, de seconde en de milliseconde beschouwd en er wordt een overeenkomend object Date geretourneerd. De waarden in deze invoerparameters worden beschouwd als waarden in de lokale tijd, niet in UTC. Met de volgende instructies wordt een object Date op middernacht op 1 januari 2000 (lokale tijd) ingesteld:

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

Vierde methode: u kunt een parameter van één tekenreeks doorgeven aan de constructor `Date()`. Vervolgens wordt geprobeerd die tekenreeks te parseren in datum- of tijdcomponenten en wordt er een overeenkomend object Date geretourneerd. Als u deze methode gebruikt, kunt u de constructor `Date()` het beste insluiten in een blok `try..catch` om eventuele fouten bij het parseren op te vangen. De constructor `Date()` accepteert een aantal verschillende tekenreeksindelingen (deze staan vermeld in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform). Met de volgende instructie wordt een nieuw object Date geïnitialiseerd met een tekenreekswaarde:

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

Als de constructor `Date()` de tekenreeksparameter niet kan parseren, treedt er geen uitzondering op. Het resulterende object Date bevat in dat geval echter wel een ongeldige datumwaarde.

Waarden voor tijdeenheden ophalen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de waarden voor verschillende tijdeenheden in een object Date extraheren met eigenschappen of methoden van de klasse Date. Elk van de volgende eigenschappen geeft u de waarde van een tijdeenheid in het object Date:

- De eigenschap `fullYear`
- De eigenschap `month`, waarvoor een numerieke notatie wordt gebruikt waarbij 0 voor januari staat en 11 voor december
- De eigenschap `date`, waarvoor het kalendergetal van de dag van de maand, een waarde tussen 1 en 31, wordt gebruikt
- De eigenschap `day`, waarvoor een numerieke notatie wordt gebruikt waarbij 0 voor zondag staat
- De eigenschap `hours`, een waarde tussen 0 en 23
- De eigenschap `minutes`
- De eigenschap `seconds`
- De eigenschap `milliseconds`

  U kunt elk van deze waarden op een aantal verschillende manieren ophalen. U kunt de maandwaarde van een object Date bijvoorbeeld op vier verschillende manieren ophalen:

- De eigenschap `month`
- De methode `getMonth()`
- De eigenschap `monthUTC`
- De methode `getMonthUTC()`

  Aangezien de vier methoden even efficiënt zijn, kunt u de benadering kiezen die het beste bij uw toepassing past.

  De bovenstaande eigenschappen vertegenwoordigen componenten van de totale datumwaarde. De eigenschap milliseconds kan bijvoorbeeld nooit een hogere waarde dan 999 hebben, omdat bij een waarde van 1000 de waarde bij seconds met 1 wordt verhoogd en de eigenschap milliseconds weer wordt ingesteld op 0.

  Als u de waarde van het object Date voor het aantal milliseconden sinds 1 januari 1970 (UTC) wilt ophalen, kunt u de methode `getTime()` gebruiken. Met diens tegenhanger, de methode `setTime()`, kunt u de waarde van een bestaand object Date wijzigen op basis van het aantal milliseconden sinds 1 januari 1970 (UTC).

Wiskundige bewerkingen op datums en tijden uitvoeren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt optel- en aftrekbewerkingen uitvoeren op datums en tijden in de klasse Date. Omdat datumwaarden intern in milliseconden worden opgeslagen, moet u andere waarden in milliseconden omzetten voordat u deze optelt bij of aftrekt van Date-objecten.

Als in uw toepassing een groot aantal wiskundige bewerkingen wordt uitgevoerd op datums en tijden, is het misschien nuttig om constanten te maken waarin veelgebruikte tijdwaarden in milliseconden worden opgeslagen, zoals hieronder:

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

Nu is het eenvoudig om wiskundige bewerkingen op datums uit te voeren met standaardtijdeenheden. Met de volgende code wordt een datumwaarde op één uur vanaf de huidige tijd ingesteld met de methoden `getTime()` en `setTime()`:

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

U kunt ook een datumwaarde instellen door een nieuw object Date te maken met één parameter voor milliseconden. Met de volgende code worden bijvoorbeeld 30 dagen bij een andere datum opgeteld om een nieuwe datum te berekenen:

```
// 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));
```

Vervolgens wordt de constante `millisecondsPerDay` met 30 vermenigvuldigd om de periode van dertig dagen te berekenen. Het resultaat wordt opgeteld bij de waarde voor `invoiceDate` en wordt gebruikt om de waarde voor `dueDate` in te stellen.

Werken met tijdzones

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt wiskundige bewerkingen op datums en tijden uitvoeren wanneer u datums wilt omzetten in een andere tijdzone. Ditzelfde doet de methode `getTimezoneOffset()`, waarmee in minuten wordt aangegeven hoeveel de tijdzone van het object Date afwijkt van UTC. Er wordt een waarde in minuten geretourneerd, omdat het verschil tussen tijdzones niet altijd een uur is. In sommige gevallen is het verschil met de volgende tijdzone een half uur.

In het volgende voorbeeld wordt het verschil in tijdzones gebruikt om een datum van de lokale tijd om te zetten in UTC. Eerst wordt de tijdzonewaarde in milliseconden berekend en vervolgens wordt de Date-waarde aangepast met die hoeveelheid:

```
// 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);
```

Tijdintervallen bepalen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u toepassingen ontwikkelt met Adobe Flash CS4 Professional, hebt u toegang tot de tijdlijn, een functie voor een evenwichtige, framegewijze voortgang door uw toepassing. In pure ActionScript-projecten moet u andere timingmechanismen gebruiken.

Lussen versus timers

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In sommige programmeertalen moet u uw eigen timingschema's maken met lusinstructies als `for` of `do..while`.

Lusinstructies worden doorgaans zo snel uitgevoerd als mogelijk is op de lokale machine, wat betekent dat de toepassing niet op elke machine even snel wordt uitgevoerd. Als voor uw toepassing een regelmatig timinginterval vereist is, moet u deze koppelen aan een actuele kalender of kloktijd. Veel toepassingen, zoals games, animaties en realtime controllers, vereisen een regelmatig, op de tijd gebaseerd tikmechanisme dat op elke machine consistent is.

De ActionScript 3.0-klasse Timer biedt een krachtige oplossing. Met het gebeurtenismodel van ActionScript 3.0 verzendt de klasse Timer timergebeurtenissen wanneer er een opgegeven tijdinterval wordt bereikt.

De klasse Timer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voor het verwerken van timingfuncties in ActionScript 3.0 kunt u het beste de klasse Timer (flash.utils.Timer) gebruiken die kan worden gebruikt om gebeurtenissen te verzenden wanneer een interval wordt bereikt.

Als u een timer wilt starten, maakt u eerst een instantie van de klasse Timer en geeft u op hoe vaak deze een timergebeurtenis moet genereren en na hoeveel keren dit moet stoppen.

Met de volgende code wordt bijvoorbeeld een instantie Timer gemaakt die gedurende 60 seconden elke seconde een gebeurtenis verzendt:

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

Elke keer dat het opgegeven interval wordt bereikt, verzendt het object Timer een object TimerEvent. Het gebeurtenistype van het object TimerEvent is `timer` (gedefinieerd door de constante `TimerEvent.TIMER`). Een object TimerEvent bevat dezelfde eigenschappen als een standaardobject Event.

Als de instantie Timer op een vast aantal intervallen is ingesteld, wordt ook de gebeurtenis `timerComplete` (gedefinieerd door de constante `TimerEvent.TIMER_COMPLETE`) verzonden wanneer het laatste interval wordt bereikt.

Hier volgt een kleine voorbeeldtoepassing waarin de klasse Timer wordt gebruikt:

```
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!");
        }
    }
}
```

Wanneer de klasse ShortTimer is gemaakt, maakt deze een instantie Timer die gedurende vijf seconden één keer per seconde tikt. Vervolgens worden twee listeners aan de timer toegevoegd: een listener die naar elke tik luistert en een listener die naar de gebeurtenis `timerComplete` luistert.

Vervolgens begint de timer te tikken en vanaf dat moment wordt de methode `onTick()` met een interval van één seconde uitgevoerd.

Met de methode `onTick()` wordt alleen de telling van het aantal tikken weergegeven. Na vijf seconden wordt de methode `onTimerComplete()` uitgevoerd en wordt u verteld dat de tijd om is.

Wanneer u dit voorbeeld uitvoert, moeten de volgende regels met een snelheid van één regel per seconde worden weergegeven in uw console of deelvenster Uitvoer:

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

Timingfuncties in het pakket flash.utils

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 bevat diverse timingfuncties die vergelijkbaar zijn met de timingfuncties in ActionScript 2.0. Dit zijn functies op pakketniveau in het pakket flash.utils, die op dezelfde manier werken als in ActionScript 2.0.

Functie	Beschrijving
`clearInterval(id:uint):void`	Hiermee wordt een opgegeven aanroep van `setInterval()` geannuleerd.
`clearTimeout(id:uint):void`	Hiermee wordt een opgegeven aanroep van `setTimeout()` geannuleerd.
`getTimer():int`	Retourneert het aantal verstreken milliseconden vanaf het moment dat Adobe® Flash® Player of Adobe® AIR™ is gestart.
`setInterval(closure:Function, delay:Number, ... arguments):uint`	Hiermee wordt een functie volgens een opgegeven interval (in milliseconden) uitgevoerd.
`setTimeout(closure:Function, delay:Number, ... arguments):uint`	Hiermee wordt een functie na een opgegeven vertraging (in milliseconden) uitgevoerd.

Deze functies zijn inbegrepen in ActionScript 3.0 voor achterwaartse compatibiliteit. Adobe raadt u niet aan deze functies te gebruiken in nieuwe ActionScript 3.0-toepassingen. Over het algemeen is het gemakkelijker en efficiënter om de klasse Timer in uw toepassingen te gebruiken.

Voorbeeld van datum en tijd: eenvoudige analoge klok

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een voorbeeld met een eenvoudige analoge klok laat deze twee concepten voor datum en tijd zien:

- De huidige datum en tijd ophalen en waarden voor de uren, minuten en seconden extraheren
- Een timer gebruiken om de snelheid van een toepassing in te stellen

  Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De bestanden van de toepassing SimpleClock vindt u in de map Samples/SimpleClock. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
SimpleClockApp.mxml of SimpleClockApp.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/simpleclock/SimpleClock.as	Het hoofdtoepassingsbestand.
com/example/programmingas3/simpleclock/AnalogClockFace.as	Hiermee worden een ronde klok met wijzers voor de uren, de minuten en de seconden getekend op basis van de tijd.

De klasse SimpleClock definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het klokvoorbeeld is eenvoudig, maar het is een goed idee om zelfs eenvoudige toepassingen goed in te delen zodat u deze later eenvoudig kunt uitbreiden. Daarom gebruikt de toepassing SimpleClock de klasse SimpleClock om taken voor het opstarten en het bijhouden van de tijd te verwerken en de klasse AnalogClockFace voor het weergeven van de tijd.

Hier volgt de code waarmee de klasse SimpleClock wordt gedefinieerd en geïnitialiseerd (in de Flash-versie breidt SimpleClock de klasse Sprite uit):

```
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;
```

De klasse heeft twee belangrijke eigenschappen:

- `De eigenschap face`, een instantie van de klasse AnalogClockFace
- `De eigenschap ticker`, een instantie van de klasse Timer

  De klasse SimpleClock gebruikt een eenvoudige standaardconstructor. Met de methode `initClock()` wordt de wijzerplaat gemaakt en wordt de instantie Timer gestart.

De wijzerplaat maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de volgende regels in de code van SimpleClock wordt de wijzerplaat gemaakt die wordt gebruikt om de tijd weer te geven:

```
/**
 * 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();
```

De grootte van de wijzerplaat kan worden doorgegeven in de methode `initClock()`. Als er geen waarde voor `faceSize` wordt doorgegeven, wordt een standaardgrootte van 200 pixels gebruikt.

De wijzerplaat wordt vervolgens door de toepassing geïnitialiseerd en toegevoegd aan het weergaveoverzicht met de methode `addChild()`, die is overgenomen van de klasse DisplayObjectContainer. Vervolgens wordt de methode `AnalogClockFace.draw()` aangeroepen om de klok één keer met de huidige tijd weer te geven.

De timer starten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de wijzerplaat is gemaakt, stelt de methode `initClock()` een timer in:

```
    // 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();
```

Eerst wordt met deze methode een instantie Timer gemaakt die één keer per seconde een gebeurtenis verzendt (elke 1000 milliseconden). Aangezien er geen tweede parameter `repeatCount` is doorgegeven aan de constructor `Timer()`, wordt de timer doorlopend herhaald.

De methode `SimpleClock.onTick()` wordt één keer per seconde uitgevoerd wanneer de gebeurtenis `timer` wordt ontvangen:

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

Met de methode `AnalogClockFace.draw()` worden alleen de wijzerplaat en de wijzers getekend.

De huidige tijd weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De meeste code in de klasse AnalogClockFace heeft betrekking op het instellen van de weergave-elementen van de klok. Wanneer de AnalogClockFace wordt geïnitialiseerd, wordt er een cirkelomtrek getekend, wordt er een numerieke tekstlabel op elke uurmarkering opgenomen en worden er drie objecten Shape gemaakt: één voor de uurwijzer, één voor de minuutwijzer en één voor de secondewijzer op de klok.

Wanneer de toepassing SimpleClock wordt uitgevoerd, wordt elke seconde de methode `AnalogClockFace.draw()` aangeroepen, op de volgende manier:

```
/**
 * 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);
}
```

Met deze methode wordt de huidige tijd in een variabele opgeslagen zodat de tijd niet kan worden gewijzigd terwijl de wijzers van de klok worden getekend. Vervolgens wordt de methode `showTime()` aangeroepen om de wijzers weer te geven, zoals hieronder wordt aangegeven:

```
/**
 * 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);
}
```

Met deze methode worden eerst de waarden voor de uren, minuten en seconden van de huidige tijd geëxtraheerd. Vervolgens worden deze waarden gebruikt om de hoek voor elke wijzer te berekenen. Aangezien de secondewijzer in 60 seconden volledig rond gaat, wordt deze elke seconde zes graden verplaatst (360/60). De minuutwijzer draait elke minuut dezelfde hoeveelheid.

De uurwijzer wordt elke minuut bijgewerkt. Deze wordt elk uur 30 graden (360/12) gedraaid en dus elke minuut een halve graad (30 graden gedeeld door 60 minuten).

Hoofdstuk 2: Werken met tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse String bevat methoden waarmee u met tekstreeksen kunt werken. Tekenreeksen zijn belangrijk bij het werken met veel objecten. De methoden die in dit hoofdstuk worden beschreven zijn nuttig voor het werken met tekenreeksen in objecten zoals TextField, StaticText, XML, Context Menu en FileReference.

Tekenreeksen zijn rijen met tekens. ActionScript 3.0 ondersteunt ASCII- en Unicode-tekens.

Meer Help-onderwerpen

String

RegExp

parseFloat()

parseInt()

Basisbeginselen van tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In programmeertaal is een tekenreeks een tekstwaarde: een reeks letters, cijfers of andere tekens die aan elkaar zijn geregen tot één waarde. Deze coderegel maakt bijvoorbeeld een variabele met het gegevenstype String en wijst een letterlijke tekenreekswaarde toe aan die variabele:

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

Zoals u in dit voorbeeld kunt zien, kunt u in ActionScript een tekenreekswaarde aangeven door tekst te omringen met dubbele of enkele aanhalingstekens. Hier volgen nog meer voorbeelden van tekenreeksen:

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

Iedere keer dat u een stuk tekst manipuleert in ActionScript, werkt u met een tekenreekswaarde. De klasse String uit ActionScript is het gegevenstype dat u kunt gebruiken om met tekstwaarden te werken. Tekenreeksinstanties worden in veel andere ActionScript-klassen vaak gebruikt voor eigenschappen, methodeparameters, enzovoort.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot tekenreeksen die in dit hoofdstuk worden gebruikt:

ASCII Een systeem voor het weergeven van schrifttekens en symbolen in computerprogramma's. Het ASCII-systeem ondersteunt het 26-letterige Engelse alfabet en een beperkte set aanvullende tekens.

Teken De kleinste eenheid van tekstgegevens (één letter of symbool).

Samenvoegen Het samenvoegen van meerdere tekenreekswaarden door een waarde te koppelen aan het einde van een andere waarde en zo een nieuwe tekenreekswaarde te maken.

Lege tekenreeks Een tekenreeks die geen tekst, witruimte of andere tekens bevat, geschreven als `""`. Een lege tekenreekswaarde verschilt van een variabele String met een waarde null. Een variabele String met de waarde null is een variabele waaraan geen instantie String is toegewezen, terwijl een lege tekenreeks een instantie heeft met een waarde die geen tekens bevat.

String Een tekstuele waarde (reeks van tekens).

Letterlijke tekenreeks Een tekenreekswaarde die expliciet in code is geschreven, geschreven als tekstwaarde die door dubbele of enkele aanhalingstekens wordt omringd.

Subtekenreeks Een tekenreeks die deel uitmaakt van een andere tekenreeks.

Unicode Een standaardsysteem voor het weergeven van schrifttekens en symbolen in computerprogramma's. Met het Unicode-systeem kunt u elk teken in elk schriftsysteem gebruiken.

Tekenreeksen maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse String wordt gebruikt om (tekstuele) tekenreeksgegevens in ActionScript 3.0 weer te geven. ActionScript-tekenreeksen ondersteunen zowel ASCII- als Unicode-tekens. De eenvoudigste manier om een tekenreeks te maken is het gebruik van een letterlijke tekenreeks. Gebruik rechte dubbele aanhalingstekens (`"`) of enkele aanhalingstekens `'`) om een letterlijke tekenreeks te declareren. De volgende twee tekenreeksen zijn bijvoorbeeld gelijk:

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

U kunt een tekenreeks ook declareren met de operator `new`, als volgt:

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

De volgende twee tekenreeksen zijn gelijk:

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

Als u enkele aanhalingstekens (`'`) wilt gebruiken binnen een letterlijke tekenreeks die met de scheidingstekens enkele aanhalingstekens (`'`) wordt gedefinieerd, gebruikt u het escape-teken backslash (`\`). Ook voor het gebruik van dubbele aanhalingstekens (`"`) binnen een letterlijke tekenreeks die met de scheidingstekens dubbele aanhalingstekens (`"`) wordt gedefinieerd, gebruikt u het escape-teken backslash (`\`). De volgende twee tekenreeksen zijn gelijk:

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

U kunt ervoor kiezen om enkele of dubbele aanhalingstekens te gebruiken op basis van enkele of dubbele aanhalingstekens die in een letterlijke tekenreeks bestaan, zoals hierna aangegeven:

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

Vergeet niet dat ActionScript onderscheid maakt tussen rechte enkele aanhalingstekens (`'`) en linkse of rechtse enkele aanhalingstekens (`' of '`). Hetzelfde geldt voor dubbele aanhalingstekens. Gebruik rechte aanhalingstekens om letterlijke tekenreeksen af te bakenen. Bij het plakken van tekst vanuit een andere bron in ActionScript, moet u erop letten dat u de juiste tekens gebruikt.

Zoals in de volgende tabel te zien is, kunt u het backslash-teken (\) gebruiken als escape-teken om andere tekens in letterlijke tekenreeksen te definiëren:

Escape-tekencombinatie	Teken
\b	Backspace
\f	Paginadoorvoer
\n	Nieuwe regel
\r	Enter
\t	Tab
\u***nnnn***	Unicode-teken met de tekencode aangeduid door het hexadecimale getal *nnnn*; \u263a is bijvoorbeeld het smiley-teken.
\\x***nn***	ASCII-teken met de tekencode aangeduid door het hexadecimale getal *nn*
\'	Enkel aanhalingsteken
\"	Dubbel aanhalingsteken
\\	Enkel backslash-teken

De eigenschap length

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elke tekenreeks heeft een eigenschap `length`, die gelijk is aan het aantal tekens in de reeks:

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

Een lege tekenreeks en een null-tekenreeks hebben beide een lengte van 0, zoals in het volgende voorbeeld wordt getoond:

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

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

Werken met tekens in tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elk teken in een tekenreeks heeft een indexpositie in de tekenreeks (een geheel getal). De indexpositie van het eerste teken is 0. In de volgende tekenreeks staat het teken `y` bijvoorbeeld op positie 0 en het teken `w` op positie 5:

```
"yellow"
```

U kunt de tekens op diverse posities in een tekenreeks onderzoeken met de methode `charAt()` en de methode `charCodeAt()`, zoals in dit voorbeeld wordt getoond:

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

Bij uitvoering van deze code wordt de volgende uitvoer geproduceerd:

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

U kunt tekencodes ook gebruiken om een tekenreeks te definiëren met de methode `fromCharCode()`, zoals in het volgende voorbeeld wordt getoond:

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

Tekenreeksen vergelijken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de volgende operatoren kunt u tekenreeksen vergelijken: `<`, `<=`, `!=`, `==`, `=>` en `>`. Deze operatoren kunnen worden gebruikt met voorwaardelijke instructies zoals `if` en `while`, zoals in het volgende voorbeeld wordt getoond:

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

Wanneer u deze operatoren met tekenreeksen gebruikt, bekijkt ActionScript de tekencodewaarde van ieder teken in de tekenreeks. Hierbij worden de tekens van links naar rechts vergeleken, zoals in het volgende voorbeeld:

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

Gebruik de operatoren `==` en `!=` om tekenreeksen met elkaar te vergelijken en om tekenreeksen met andere typen objecten te vergelijken, zoals in het volgende voorbeeld wordt getoond:

```
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
```

Tekenreeksrepresentaties van andere objecten verkrijgen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een tekenreeksrepresentatie verkrijgen voor elk soort object. Alle objecten hebben een methode `toString()` voor dit doeleinde:

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

Wanneer u de samenvoegingsoperator + gebruikt met een combinatie van objecten String en objecten die geen tekenreeksen zijn, hoeft u de methode `toString()` niet te gebruiken. Zie de volgende sectie voor details over samenvoeging.

De algemene functie `String()` retourneert dezelfde waarde voor een bepaald object als de waarde die wordt geretourneerd door het object dat de methode `toString()` aanroept.

Tekenreeksen samenvoegen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u tekenreeksen samenvoegt, maakt u één tekenreeks van twee afzonderlijke tekenreeksen. U kunt bijvoorbeeld de operator + gebruiken om twee tekenreeksen samen te voegen:

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

U kunt ook de operator += gebruiken om hetzelfde resultaat te produceren, zoals in het volgende voorbeeld wordt getoond:

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

Bovendien bevat de klasse String een methode `concat()`, die als volgt kan worden gebruikt:

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

Wanneer u de operator + (of de operator +=) gebruikt met een object String en een object dat *geen* tekenreeks is, zet ActionScript het object dat geen tekenreeks is automatisch om in een object String om de expressie te evalueren, zoals in dit voorbeeld wordt getoond:

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

U kunt echter ronde haakjes gebruiken om te groeperen om context te leveren voor de operator +, zoals in het volgende voorbeeld wordt getoond:

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

Subtekenreeksen en patronen zoeken in tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Subtekenreeksen zijn opeenvolgende tekens binnen een tekenreeks. De tekenreeks `"abc"` heeft bijvoorbeeld de volgende subtekenreeksen: `""`, `"a"`, `"ab"`, `"abc"`, `"b"`, `"bc"` en `"c"`. Met ActionScript-methoden kunt u zoeken naar subtekenreeksen van een tekenreeks.

Patronen worden in ActionScript gedefinieerd door tekenreeksen of door reguliere expressies. De volgende reguliere expressie definieert bijvoorbeeld een specifiek patroon: de letters A, B en C gevolgd door een cijferteken (de slashes zijn reguliere-expressiescheidingstekens):

```
/ABC\d/
```

ActionScript bevat methoden voor het zoeken van patronen in tekenreeksen en voor het vervangen van gevonden overeenkomsten met vervangende subtekenreeksen. Deze methoden worden beschreven in de volgende secties.

Reguliere expressies kunnen ingewikkelde patronen definiëren. Zie “Reguliere expressies gebruiken” op pagina 81 voor meer informatie over reguliere expressies.

Een subtekenreeks zoeken op tekenpositie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methoden `substr()` en `substring()` zijn vergelijkbaar. Beide methoden retourneren een subtekenreeks van een tekenreeks. Beide hebben twee parameters. In beide methoden is de eerste parameter de positie van het teken aan het begin van de tekenreeks. In de methode `substr()` is de tweede parameter echter de *lengte* van de subtekenreeks die wordt geretourneerd; in de methode `substring()` is de tweede parameter de positie van het teken aan het *einde* van de subtekenreeks (die niet is opgenomen in de geretourneerde tekenreeks). Dit voorbeeld laat het verschil zien tussen deze twee methoden:

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

De methode `slice()` werkt ongeveer op dezelfde manier als de methode `substring()`. Met twee niet-negatieve gehele getallen als parameters werken ze exact gelijk. De methode `slice()` kan echter negatieve gehele getallen als parameters hebben. In dit geval wordt de tekenpositie vanuit het einde van de tekenreeks gehaald, zoals in het volgende voorbeeld wordt getoond:

```
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
```

U kunt niet-negatieve en negatieve gehele getallen combineren als parameters van de methode `slice()`.

De tekenpositie zoeken van een overeenkomende subtekenreeks

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methoden `indexOf()` en `lastIndexOf()` kunt u overeenkomende subtekenreeksen opzoeken in een tekenreeks, zoals in het volgende voorbeeld wordt getoond:

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

De methode `indexOf()` maakt onderscheid tussen hoofd- en kleine letters.

U kunt als volgt een tweede parameter opgeven om de indexpositie aan te geven in de tekenreeks van waaruit de zoekopdracht wordt gestart:

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

De methode `lastIndexOf()` zoekt naar de laatste instantie van een subtekenreeks in de tekenreeks:

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

Bij het gebruik van een tweede parameter bij de methode `lastIndexOf()`, wordt de zoekopdracht in achterwaartse richting (van rechts naar links) uitgevoerd vanaf de desbetreffende indexpositie in de tekenreeks.

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

Een array subtekenreeksen maken met behulp van een scheidingsteken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `split()` kunt u een array subtekenreeksen maken met behulp van een scheidingsteken. U kunt bijvoorbeeld een tekenreeks met door komma's of tabs gescheiden waarden segmenteren in meerdere tekenreeksen.

Het volgende voorbeeld laat zien hoe u een array kunt opsplitsen in subtekenreeksen met de ampersand (&) als scheidingsteken:

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

De tweede parameter van de methode `split()`, die optioneel is, definieert de maximumgrootte van de array die wordt geretourneerd.

U kunt ook een reguliere expressie gebruiken als scheidingsteken:

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

Zie "Reguliere expressies gebruiken" op pagina 81 en de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Patronen in tekenreeksen zoeken en subtekenreeksen vervangen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse String bevat de volgende methoden voor het werken met patronen in tekenreeksen:

- Gebruik de methoden `match()` en `search()` om subtekenreeksen te zoeken die overeenkomen met een patroon.
- Gebruik de methode `replace()` om subtekenreeksen te zoeken die overeenkomen met een patroon en deze subtekenreeksen te vervangen met een opgegeven subtekenreeks.

Deze methoden worden beschreven in de volgende secties.

U kunt tekenreeksen of reguliere expressies gebruiken om patronen die in deze methoden worden gebruikt, te definiëren. Zie "Reguliere expressies gebruiken" op pagina 81 voor meer informatie over reguliere expressies.

Overeenkomende subtekenreeksen zoeken

De methode `search()` retourneert de indexpositie van de eerste subtekenreeks die overeenkomt met een gegeven patroon, zoals in dit voorbeeld wordt getoond:

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

U kunt ook reguliere expressies gebruiken om het patroon waarmee moet worden overeengekomen te definiëren, zoals in dit voorbeeld wordt getoond:

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

De uitvoer van de methode `trace()` is 0, omdat het eerste teken in de tekenreeks indexpositie 0 is. De markering `i` wordt ingesteld in de reguliere expressie, zodat de zoekopdracht geen onderscheid maakt tussen hoofd- en kleine letters.

De methode `search()` vindt slechts een overeenkomst en retourneert de beginindexpositie, zelfs als de markering `g` (global) is ingesteld in de reguliere expressie.

In het volgende voorbeeld wordt een ingewikkeldere reguliere expressie getoond, die overeenkomt met een tekenreeks tussen dubbele aanhalingstekens:

```
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.)
```

De methode `match()` werkt ongeveer op dezelfde manier. De methode zoekt naar een overeenkomende subtekenreeks. Wanneer u echter de algemene markering gebruikt in een reguliere-expressiepatroon, zoals in het volgende voorbeeld, retourneert `match()` een array van overeenkomende subtekenreeksen:

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

De array `results` wordt op het volgende ingesteld:

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

Zie "Reguliere expressies gebruiken" op pagina 81 voor meer informatie over reguliere expressies.

Overeenkomende subtekenreeksen vervangen

Met de methode `replace()` kunt u in een tekenreeks zoeken naar een opgegeven patroon en overeenkomende items vervangen door de opgegeven vervangende tekenreeks, zoals in het volgende voorbeeld wordt getoond:

```
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.
```

In dit voorbeeld wordt voor de overeenkomende tekenreeksen geen onderscheid gemaakt tussen hoofd- en kleine letters, omdat de markering `i` (`ignoreCase`) is ingesteld in de reguliere expressie, en meerdere overeenkomende items worden vervangen, omdat de markering `g` (`global`) is ingesteld. Zie "Reguliere expressies gebruiken" op pagina 81 voor meer informatie.

U kunt de volgende `$`-vervangingscodes in de vervangende tekenreeks opnemen: De vervangende tekst die in de volgende tabel wordt getoond, wordt ingevoegd op de plaats van de vervangingscode `$`:

$-code	**Vervangingstekst**
`$$`	$
`$&`	De overeenkomende subtekenreeks.
`$``	Het gedeelte van de tekenreeks dat voorafgaat aan de overeenkomende subtekenreeks. In deze code wordt het rechte enkele aanhalingsteken links (`) gebruikt, niet het rechte enkele aanhalingsteken (') of het gekrulde enkele aanhalingsteken links (‘).
`$'`	Het gedeelte van de tekenreeks dat volgt na de overeenkomende subtekenreeks. Deze code gebruikt rechte enkele aanhalingstekens (').
$n	De *n*-de vastgelegde overeenkomende groep tussen ronde haakjes, waarbij n een cijfer (1-9) is en $n niet wordt gevolgd door een decimaal cijfer.
$nn	De *nn*-de vastgelegde overeenkomende groep tussen ronde haakjes, waarbij *nn* een tweecijferig decimaal getal is (01–99). Als de *nn*-de vastlegging niet gedefinieerd is, is de vervangende tekst een lege tekenreeks.

Hierna volgt een voorbeeld van de vervangingscodes `$2` en `$1`, die de eerste en tweede vastgelegde overeenkomende groep vertegenwoordigen:

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

U kunt ook een functie gebruiken als tweede parameter van de methode `replace()`. De overeenkomende tekst is vervangen door de geretourneerde waarde van de functie.

```
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;
}
```

Wanneer u een functie opgeeft als tweede parameter van de methode `replace()`, worden de volgende argumenten doorgegeven aan de functie:

- Het overeenkomende gedeelte van de tekenreeks.
- Alle vastleggende overeenkomende groepen tussen ronde haakjes. Het aantal doorgegeven argumenten hangt af van het aantal overeenkomsten tussen ronde haakjes. U kunt het aantal overeenkomsten tussen ronde haakjes vaststellen door `arguments.length - 3` binnen de functiecode te controleren.
- De indexpositie in de tekenreeks waar de overeenkomst begint.
- De volledige tekenreeks.

Tekenreeksen omzetten tussen hoofdletters en kleine letters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals in het volgende voorbeeld wordt getoond, zetten de methoden `toLowerCase()` en `toUpperCase()` alfabetische tekens in de tekenreeks om in respectievelijk kleine letters en hoofdletters:

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

Na uitvoering van deze methoden, blijft de brontekenreeks ongewijzigd. U kunt met de volgende code de brontekenreeks transformeren:

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

Deze methoden werken met uitgebreide tekens, niet alleen a–z en A–Z:

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

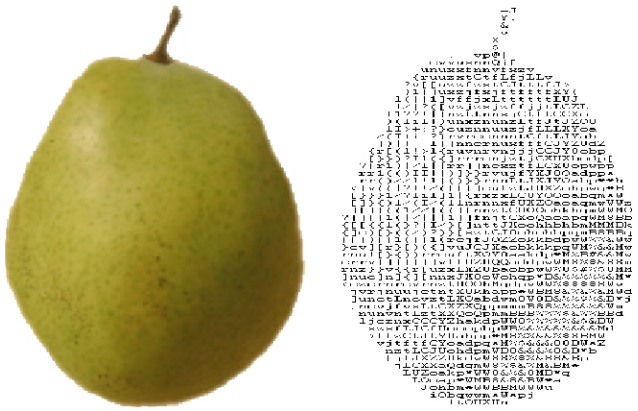
Voorbeeld van strings: ASCII-afbeeldingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Dit voorbeeld van een ASCII-tekening laat een aantal kenmerken zien van het werken met de klasse String in ActionScript 3.0, waaronder het volgende:

- De methode `split()` van de klasse String wordt gebruikt om waarden te extraheren van een door tekens gescheiden tekenreeks (afbeeldingsinformatie in een door tabs gescheiden tekstbestand).
- U kunt verschillende tekenreeksmanipulatietechnieken, waaronder `split()`, samenvoeging en het extraheren van een deel van de tekenreeks met `substring()` en `substr()`, gebruiken om de eerste letter van ieder woord in de titels van afbeeldingen een hoofdletter te geven.
- De methode `getCharAt()` wordt gebruikt om één teken uit een tekenreeks op te halen (om vast te stellen welk ASCII-teken overeenkomt met een bitmapwaarde voor grijstinten).
- Tekenreekssamenvoeging wordt gebruikt om de weergave van een ASCII-tekening met één teken tegelijk op te bouwen.

De term *ASCII-tekening* verwijst naar de tekstweergave van een afbeelding, waarbij de afbeelding in kaart wordt gebracht door een raster van lettertypetekens met vaste tekenbreedte, zoals Courier New-tekens. De volgende afbeelding toont een voorbeeld van een ASCII-tekening die door de toepassing wordt geproduceerd:



De ASCII-tekening van de afbeelding wordt rechts weergegeven.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van ASCIIArt vindt u in de map Samples/ASCIIArt. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
AsciiArtApp.mxml of AsciiArtApp.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML)
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	De klasse met de hoofdfunctionaliteit van de toepassing, waaronder het extraheren van metagegevens van een afbeelding uit een tekstbestand, het laden van de afbeeldingen en het beheren van het omzettingsproces van afbeelding in tekst.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	Een klasse die de methode `parseBitmapData()` biedt voor het omzetten van afbeeldingsgegevens in een versie String.
com/example/programmingas3/asciiArt/Image.as	Een klasse die een geladen bitmapafbeelding weergeeft.
com/example/programmingas3/asciiArt/ImageInfo.as	Een klasse die metagegevens vertegenwoordigt voor een ASCII-tekening (zoals titel, afbeeldingsbestands-URL, enzovoort).
image/	Een map met de afbeeldingen die door de toepassing worden gebruikt.
txt/ImageData.txt	Een door tabs gescheiden tekstbestand met informatie over bestanden die door de toepassing moeten worden geladen.

Door tabs gescheiden waarden extraheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In dit voorbeeld wordt de standaardmethode gebruikt om toepassingsgegevens apart van de toepassing zelf op te slaan. Wanneer gegevens worden gewijzigd (bijvoorbeeld, als een andere afbeelding wordt toegevoegd of de titel van een afbeelding wijzigt), is het niet nodig het SWF-bestand opnieuw te maken. In dit geval worden de metagegevens van de afbeelding, inclusief de titel van de afbeelding, de URL van het daadwerkelijke afbeeldingsbestand en een aantal waarden die worden gebruikt om de afbeelding te manipuleren, opgeslagen in een tekstbestand (het bestand txt/ImageData.txt in het project). De inhoud van het tekstbestand is als volgt:

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

Het bestand gebruikt een specifieke door tabs gescheiden indeling. De eerste regel (rij) is een rij met koptekst. De resterende regels bevatten de volgende gegevens voor iedere te laden bitmap:

- De bestandsnaam van de bitmap.
- De weergavenaam van de bitmap.
- De bitmapdrempelwaarden voor wit en voor zwart. Dit zijn hexadecimale waarden waarboven en waaronder een pixel wordt beschouwd als compleet wit of compleet zwart.

Zodra de toepassing start, wordt de klasse AsciiArtBuilder geladen, waarmee de inhoud van het tekstbestand wordt geparseerd. Zo wordt de 'stapel' met afbeeldingen gemaakt die wordt weergegeven. Dit gebeurt met behulp van de volgende code uit de methode `parseImageInfo()` van de klasse 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);
    }
}
```

De gehele inhoud van het tekstbestand wordt opgenomen in één instantie String, de eigenschap `_imageInfoLoader.data`. Wanneer u de methode `split()` met het teken voor een nieuwe regel (`"\n"`) gebruikt als parameter, wordt de instantie String verdeeld in een Array (`lines`) waarvan de elementen de individuele regels van het tekstbestand zijn. Vervolgens gebruikt de code een lus om met iedere regel te werken (behalve de eerste, omdat die alleen kopteksten bevat en geen daadwerkelijke inhoud). Binnen de lus wordt de methode `split()` nogmaals gebruikt om de inhoud van de ene regel te verdelen in een set waarden (het object Array met de naam `imageProperties`). De parameter die in dit geval bij de methode `split()` wordt gebruikt is het tab-teken (`"\t"`), omdat de waarden in iedere regel door tab-tekens worden afgebakend.

Methoden String gebruiken om afbeeldingstitels te normaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een van de ontwerpbesluiten voor deze toepassing is dat alle afbeeldingstitels worden weergegeven met een standaardindeling, met de eerste letter van ieder woord als hoofdletter (behalve een paar woorden die normaal gesproken niet met een hoofdletter worden geschreven in titels). In plaats van aan te nemen dat het tekstbestand titels bevat die op de juiste manier zijn opgemaakt, maakt de toepassing de titels op terwijl ze uit het tekstbestand worden geëxtraheerd.

In het vorige codevoorbeeld wordt, als onderdeel van het extraheren van waarden voor metagegevens van afzonderlijke afbeeldingen, de volgende coderegel gebruikt:

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

In deze code wordt de titel van de afbeelding van het tekstbestand aan de methode `normalizeTitle()` doorgegeven voordat deze in het object ImageInfo wordt opgeslagen:

```
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(" ");
}
```

Bij deze methode wordt de methode `split()` gebruikt om de titel in afzonderlijke woorden te verdelen (gescheiden door het spatieteken), waarna elk woord wordt doorgegeven aan de methode `capitalizeFirstLetter()`. Vervolgens wordt de methode `join()` van de klasse Array gebruikt om de woorden te combineren, zodat ze weer één tekenreeks vormen.

Met de methode `capitalizeFirstLetter()` wordt dus van de eerste letter van elk woord een hoofdletter gemaakt:

```
/**
 * 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;
}
```

Voor het Engels geldt dat het eerste teken van woorden in een titel *geen* hoofdletter krijgt als het een van de volgende woorden betreft: 'and', 'the', 'in', 'an', 'or', 'at', 'of' of 'a'. (Dit is een vereenvoudigde versie van de regels.) Voor het uitvoeren van deze logica, gebruikt de code eerst een instructie `switch` om te controleren of het woord een van de woorden is die geen hoofdletter mogen krijgen. Als dat het geval is, springt de code buiten de instructie `switch`. Als het woord wel een hoofdletter moet krijgen, worden de volgende stappen doorlopen:

1. De eerste letter van het woord wordt met `substr(0, 1)` opgehaald, waarbij een subtekenreeks wordt opgehaald dat met een teken op index 0 begint (de eerste letter in de tekenreeks, zoals aangegeven door de eerste parameter `0`). De subtekenreeks is één teken lang (aangeduid door de tweede parameter `1`).
2. Dat teken krijgt een hoofdletter met de methode `toUpperCase()`.
3. De overige tekens van het oorspronkelijke woord worden met `substring(1)` opgehaald, waarbij een subtekenreeks wordt opgehaald dat op index 1 begint (de tweede letter) tot het einde van de tekenreeks (aangeduid door de tweede parameter van de methode `substring()` weg te laten).
4. Het uiteindelijke woord wordt gemaakt door de eerste letter die net een hoofdletter heeft gekregen, te combineren met de overige letters via tekenreekssamenvoeging: `firstLetter + otherLetters`.

ASCII-tekeningtekst genereren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse BitmapToAsciiConverter biedt functionaliteit voor het omzetten van een bitmapafbeelding in de ASCII-tekstrepresentatie ervan. Dit proces wordt uitgevoerd door de methode `parseBitmapData()` en wordt hier gedeeltelijk weergegeven:

```
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;
```

Deze code definieert eerst een instantie String met de naam `result`, die wordt gebruikt om de ASCII-versie van de bitmapafbeelding te maken. Vervolgens worden de afzonderlijke pixels van de bronbitmapafbeelding doorlopen. Bij het gebruik van verschillende kleurmanipulatietechnieken (hier weggelaten voor bondigheid), zet de code de rode, groene en blauwe waarden van een afzonderlijke pixel om in één grijsschaalwaarde (een getal van 0 tot 255). Vervolgens wordt de waarde omgezet in een waarde in de 0-63-schaal door de waarde door 4 te delen (zoals in het codevoorbeeld wordt getoond). Deze bewerkte waarde wordt opgeslagen in de variabele `index`. (De 0-63-schaal wordt gebruikt, omdat het 'palet' van beschikbare ASCII-tekens die deze toepassing gebruikt, 64 tekens bevat.) Het palet met tekens wordt gedefinieerd als een instantie String in de klasse 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><+_~-;,. ";
```

Omdat de variabele `index` bepaalt welk ASCII-teken in het palet overeenkomt met de huidige pixel in de bitmapafbeelding, wordt dat teken opgehaald uit het `palet` String via de methode `charAt()`. Vervolgens wordt het teken toegevoegd aan de String-instantie `result` via de toewijzingsoperator voor samenvoegen (`+=`). Verder wordt aan het eind van elke rij met pixels een teken voor een nieuwe regel samengevoegd met het eind van de String-instantie `result`, zodat tekstomloop wordt afgedwongen voor de regel en een nieuwe regel met 'tekenpixels' wordt gemaakt.

Hoofdstuk 3: Werken met arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met een array kunt u meerdere waarden opslaan in één gegevensstructuur. U kunt eenvoudige geïndexeerde arrays gebruiken waarin waarden worden opgeslagen met behulp van een vaste ordinale index met gehele getallen of complexe associatieve arrays waarin waarden worden opgeslagen met behulp van willekeurige sleutels. Een array kan ook multidimensionaal zijn: met elementen die zelf een array zijn. Tot slot kunt u een Vector gebruiken voor een array waarvan alle elementen instanties van hetzelfde gegevenstype zijn.

Meer Help-onderwerpen

Array

Vector

Basisbeginselen van arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Vaak zult u tijdens het programmeren met een reeks items moeten werken in plaats van met één enkel object. Zo hebt u in een muziekspeler mogelijk een lijst met nummers die in de wachtrij staan om te worden afgespeeld. U wilt natuurlijk niet voor elk nummer in die lijst een afzonderlijke variabele maken. Het is wenselijker als u alle nummerobjecten in een bundel kunt plaatsen en deze als groep kunt bewerken.

Een array is een programmeerelement dat fungeert als container voor een set items, zoals een lijst met nummers. Meestal zijn alle items van een array een instantie van dezelfde klasse, maar voor ActionScript is dat niet echt nodig. De afzonderlijke items in een array worden de *elementen* van de array genoemd. Een array kan worden beschouwd als een ladekast voor variabelen. Variabelen kunnen als element aan de array worden toegevoegd, zoals u een dossier in de ladekast legt. U kunt met de array werken als met een enkele variabele (net zoals u de lade in zijn geheel kunt verplaatsen). U kunt deze variabelen als een groep bewerken (zoals u de mappen één voor één kunt doorzoeken naar een stukje informatie). U kunt ze ook afzonderlijk benaderen (zoals u de lade kunt openen om één enkele map te selecteren).

Stel dat u een muziekspeler wilt maken waarin het mogelijk is om meerdere nummers te selecteren en aan een afspeellijst toe te voegen. In uw ActionScript-code werkt u met de methode `addSongsToPlaylist()`, waarbij één array wordt geaccepteerd als parameter. Het maakt niet uit hoeveel nummers u aan de afspeellijst wilt toevoegen (een paar, veel, misschien maar één), u hoeft de methode `addSongsToPlaylist()` maar één keer op te roepen en de array met de nummerobjecten wordt doorgegeven aan de methode. Binnen de methode `addSongsToPlaylist()` kunt u een lus gebruiken om de elementen van de array (de nummers) één voor één te doorlopen en ze concreet aan de afspeellijst toe te voegen.

Het meest voorkomende type ActionScript-array is een *geïndexeerde array*. In een geïndexeerde array wordt elk item opgeslagen in een genummerd vak (de *index* genaamd). Items kunnen middels dat nummer worden benaderd, net als een adres. Een geïndexeerde array is voor de meeste programmeerbehoeften een prima oplossing. De klasse Array is een veelvoorkomende klasse die wordt gebruikt om een geïndexeerde klasse voor te stellen.

Een geïndexeerde array wordt dikwijls gebruikt om meerdere items van hetzelfde type op te slaan (objecten die instanties zijn van dezelfde klasse). De klasse Array is niet bedoeld om het type items dat deze bevat, te beperken. De klasse Vector is een type geïndexeerde array waarin alle items in één enkele array van hetzelfde type zijn. Het gebruik van een Vector-instantie in plaats van een Array-instantie biedt prestatieverbeteringen en andere voordelen. De klasse Vector is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5.

Een speciale toepassing van een geïndexeerde array is een *multidimensionale array*. Een multidimensionale array is een geïndexeerde array waarvan de elementen geïndexeerde arrays zijn (die zelf weer andere elementen bevatten).

Een ander type array is een *associatieve array*, waarin een *sleutel*reeks in plaats van een numerieke index wordt gebruikt om de verschillende elementen te identificeren. Tot slot bevat ActionScript 3.0 ook nog de klasse Dictionary, die een *woordenboek* voorstelt. Een woordenboek is een array waarbij u een willekeurig type object als sleutel kunt gebruiken om de verschillende elementen van elkaar te onderscheiden.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen die u tegenkomt bij het programmeren van array- en vectorafhandelingsroutines.

Array een object dat fungeert als container waarin meerdere objecten worden samengebracht.

Arraytoegangoperator ([]) Een paar vierkante haakjes rond een index of toets die een array-element aangeven. Deze syntaxis wordt achter de naam van een arrayvariabele gebruikt als u een enkel element van de array in plaats van de hele array wilt opgeven.

Associërende array Een array die tekenreekstoetsen gebruikt om individuele elementen aangeeft.

Basistype Het gegevenstype van objecten dat in een vectorinstantie mag worden opgeslagen.

Woordenboek Een array waarvan de items bestaan uit objectparen, die uit een key en een waarde bestaan. Voor de identificatie van een afzonderlijk element wordt de sleutel gebruikt in plaats van een numerieke index.

Element een afzonderlijk item in een array.

Index het numerieke 'adres' dat wordt gebruikt voor de identificatie van een afzonderlijk element in een geïndexeerde array.

Geïndexeerde array Het standaardtype array dat elk element in een genummerde locatie opslaat en dat het nummer (index) gebruikt om individuele elementen te identificeren.

Key De tekenreeks of het object dat wordt gebruikt om één element in een geassocieerde array of in een woordenboek te identificeren.

Multidimensionale array Een array bevat items die arrays zijn in plaats van enkele waarden.

T De standaardconventie die in deze documentatie wordt gebruikt om het basistype van een Vectorinstantie weer te geven, bij elk basistype. De T-conventie wordt gebruikt om de naam van een klasse weer te geven, zoals in de beschrijving van de parameter Type wordt weergegeven. (“T” betekent "type", in dit geval "gegevenstype.").

Type parameter De syntaxis die wordt gebruikt met de klassennaam Vector om het basistype van de Vector aan te geven (het gegevenstype van de objecten die erin worden opgeslagen). De syntaxis bestaat uit een punt (.), gevolgd door de naam van het gegevenstype tussen vierkante haken (<>). Bij elkaar ziet het er als volgt uit: `Vector.<T>`. In deze documentatie wordt de klasse die in de parameter Type wordt opgegeven, gewoonlijk voorgesteld als `T`.

Vector een type array waarvan de elementen allemaal instanties van hetzelfde gegevenstype zijn.

Geïndexeerde arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Geïndexeerde arrays slaan een reeks van een of meer waarden op, die op zo'n manier worden geordend dat elke waarde kan worden benaderd met een geheel-getalwaarde zonder teken. De eerste index is altijd het getal 0 en de index neemt met 1 toe voor elk achtereenvolgend element dat u toevoegt aan de array. In ActionScript 3.0 worden twee klassen als geïndexeerde arrays gebruikt: de klasse Array en de klasse Vector.

Geïndexeerde arrays gebruiken voor het indexnummer een 32-bits geheel getal zonder teken. De maximale grootte van een geïndexeerde array is 2^{32} - 1 of 4.294.967.295. Een poging om een array te maken die groter is dan dit maximum, resulteert in een uitvoeringsfout.

Als u één element van een geïndexeerde array wilt benaderen, gebruikt u de operator voor arraytoegang (`[]`) om de indexpositie op te geven van het element dat u wilt benaderen. De volgende code bijvoorbeeld wijst het eerste element aan (het element op index 0) in de geïndexeerde array `songTitles`:

```
songTitles[0]
```

De combinatie van de naam van de arrayvariabele gevolgd door de index tussen vierkante haken werkt als één identificatie. (Met andere woorden, deze kan op dezelfde wijze als een variabelennaam worden gebruikt.) U kunt een waarde aan een geïndexeerd arrayelement toewijzen door de naam en index aan de linkerzijde van een toewijzingsinstructie te plaatsen:

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

Evenzo kunt u de waarde van een geïndexeerd arrayelement ophalen door de naam en index aan de rechterzijde van een toewijzingsinstructie te plaatsen:

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

U kunt ook een variabele tussen vierkante haken plaatsen in plaats van een expliciete waarde op te geven. (De variabele moet een niet-negatieve gehele waarde zijn, zoals een uint, een positieve int of een positieve gehele instantie van Number.) Deze techniek wordt veelvuldig gebruikt om de elementen in een geïndexeerde array te doorlopen en met enkele of alle elementen een bewerking uit te voeren. In het volgende stukje programmacode wordt deze techniek gedemonstreerd. In de code wordt een lus gebruikt om elke waarde in het Array-object `oddNumbers` te benaderen. De instructie `trace()` wordt gebruikt om elke waarde af te drukken in de vorm 'oddNumber[*index*] = *waarde*':

```
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());
}
```

De klasse Array

Het eerste type geïndexeerde array is de klasse Array. Een Array-instantie kan een waarde van een willekeurig gegevenstype bevatten. Hetzelfde Array-object kan objecten van verschillende gegevenstypen bevatten. Zo kan een enkele Array-instantie een String-waarde op index 0 hebben, een Number-instantie op index 1 en een XML-object op index 2.

De klasse Vector

Een ander type geïndexeerde array in ActionScript 3.0 is de klasse Vector. Een Vector-instantie is een *array met type*. Dat betekent dat alle elementen in een Vector-instantie allemaal hetzelfde gegevenstype hebben.

Opmerking: *De klasse Vector is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5.*

Wanneer u een Vector-variabele declareert of een Vector-object instantieert, geeft u expliciet het gegevenstype op van de objecten die de vector kan bevatten. Het opgegeven gegevenstype wordt het *basistype* van de vector genoemd. Tijdens de runtime en het compileren (in strikte modus) wordt elke code gecontroleerd die de waarde van een Vector-element instelt of een waarde uit een vector ophaalt. Als het gegevenstype van het object dat wordt toegevoegd of opgehaald, niet overeenkomt met het basistype van de vector, treedt een fout op.

De vectorklasse heeft naast de gegevenstypebeperking ook andere beperkingen die deze van de klasse Array onderscheiden:

- Een vector is een dichte array. Een Array-object kan waarden bevatten op de indices 0 en 7, zelfs als de indices 1 tot en met 6 geen waarde bevatten. Een vector moet echter op elke index een waarde (of `null`) hebben.
- Een vector kan optioneel een vaste lengte hebben. Dat betekent dat het aantal elementen dat de vector kan bevatten, niet kan veranderen.
- Toegang tot de elementen van een vector wordt met begrenzingen gecontroleerd. U kunt nooit een waarde lezen uit een index die groter is dan het laatste element (`length` - 1). U kunt nooit een waarde instellen met een index die hoger is dan de huidige laatste index. (Met andere woorden, u kunt alleen een waarde instellen op een bestaande index of op index `[length]`.)

Een Vector-instantie heeft, door zijn beperkingen, drie belangrijke voordelen ten opzichte van een Array-instantie waarvan de elementen allemaal van één klasse zijn:

- Prestatie: toegang tot en iteratie van arrayelementen zijn veel sneller wanneer u een Vector-instantie gebruikt dan met een Array-instantie.
- Veiligheid van type: in de strikte modus kan de compiler gegevenstypefouten onderscheppen. Voorbeelden van dergelijke fouten zijn het toewijzen van een waarde van een verkeerd gegevenstype aan een vector, of het verwachten van het verkeerde gegevenstype wanneer een waarde uit een vector wordt gelezen. Tijdens de runtime worden gegevenstypen eveneens gecontroleerd wanneer ze worden toegevoegd aan of gelezen uit een Vector-object. Wanneer u echter de methode `push()` of `unshift()` gebruikt om waarden aan een vector toe te voegen, worden de gegevenstypen van het argument niet tijdens het compileren gecontroleerd. Bij gebruik van deze methoden worden de waarden wel tijdens de runtime gecontroleerd.
- Betrouwbaarheid: door bereikcontroles tijdens de runtime (of vaste-lengtecontroles) is de betrouwbaarheid aanzienlijk groter dan bij arrays.

Op de extra beperkingen en voordelen na, lijkt de klasse Vector verder veel op de klasse Array. De eigenschappen en methoden van een Vector-object zijn vergelijkbaar - voornamelijk identiek - met de eigenschappen en methoden van een array. In de meeste gevallen waarin u een array zou gebruiken waarin alle elementen hetzelfde gegevenstype hebben, heeft een Vector-instantie de voorkeur.

Arrays maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn diverse technieken waarmee u een Array- of een Vector-instantie kunt maken. De technieken voor het maken van elk type array verschillen echter enigszins.

Array-instanties maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een Array-object maken door de `Array()`-constructor op te roepen of door de literale Array-syntaxis te gebruiken.

De constructorfunctie voor `Array()` kan op drie manieren worden gebruikt. Ten eerste: als u de constructor aanroept zonder argumenten, krijgt u een lege array. Met de eigenschap `length` van de klasse Array kunt u controleren of de array geen elementen heeft. Met de volgende code wordt bijvoorbeeld de `Array ()`-constructor opgeroepen zonder argumenten:

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

Ten tweede: als u als enige parameter voor de `Array()`-constructor een nummer gebruikt, wordt een array met die lengte gemaakt, waarbij de waarde van elk element is ingesteld op `undefined`. Het argument moet een geheel getal zonder teken zijn tussen de waarden 0 en 4.294.967.295. Met de volgende code wordt bijvoorbeeld de `Array()`-constructor opgeroepen met één numeriek argument:

```
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
```

Ten derde: als u de constructor oproept en een lijst met elementen doorgeeft als parameter, wordt een array gemaakt met elementen die overeenstemmen met de verschillende parameters. Met de volgende code worden drie argumenten aan de `Array()`-constructor doorgegeven:

```
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
```

U kunt ook arrays maken met Array-literalen. Array-literalen kunnen rechtstreeks aan een arrayvariabele worden toegewezen, zoals in het volgende voorbeeld wordt geïllustreerd:

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

Vector-instanties maken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U maakt een Vector-instantie door de constructor`Vector.<T>()`-constructor. U kunt ook een vector maken door de globale functie `Vector.<T>()`-constructor. Deze functie converteert een opgegeven object in een Vector-instantie. In Flash Professional CS5 of hoger, Flash Builder 4 of hoger en Flex 4 of hoger kunt u ook een Vector-instantie maken door de letterlijke Vector-syntaxis te gebruiken.

Telkens wanneer u een Vector-variabele declareert (of een parameter voor de Vector-methode of een resultaattype voor de methode), geeft u het basistype van de Vector-variabele op. U geeft het basistype ook op wanneer u een Vector-instantie maakt door de constructor `Vector.<T>()`-constructor. Anders gezegd, telkens wanneer u de term `Vector` in ActionScript gebruikt, gaat deze vergezeld van een basistype.

U geeft het basistype van de vector op met de syntaxis voor de parameter type. Het parametertype staat direct achter het woord `Vector` in de code. Deze bestaat uit een punt (.), gevolgd door de naam van de basisklasse tussen vierkante haken (`<>`), zoals in dit voorbeeld:

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

In de eerste regel van het voorbeeld is de variabele `v` gedeclareerd als een `Vector.<String>`-instantie. Met andere woorden, deze geeft een geïndexeerde array aan die alleen String-instanties kan bevatten. Op de tweede regel wordt de `Vector()`-constructor opgeroepen om een instantie van hetzelfde Vector-type te maken (dus een vector waarvan de elementen allemaal String-objecten zijn). Hier wordt dat object toegewezen aan `v`.

De Vector.<T>()-constructor gebruiken

Als u de constructor `Vector.<T>()` zonder argumenten gebruikt, wordt een lege Vector-instantie gemaakt. U kunt testen of een vector leeg is door de eigenschap `length` van de vector te controleren. Met de volgende code wordt bijvoorbeeld de constructor `Vector.<T>()` opgeroepen zonder argumenten:

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

Als u van tevoren weet hoeveel elementen een vector nodig heeft, kunt u het aantal elementen in de vector van tevoren definiëren. Als u een vector met een bepaald aantal elementen wilt maken, geeft u het aantal elementen als eerste parameter (de parameter `length`) door. Omdat Vector-elementen niet leeg mogen zijn, worden alle elementen gevuld met instanties van het basistype. Als het basistype een referentietype is dat `null`-waarden toestaat, bevatten de elementen allemaal `null`-waarden. Anders bevatten de elementen allemaal de standaardwaarde voor de klasse. Zo mag een uint-variabele niet `null` zijn. Daarom wordt de vector `ages` in de volgende voorbeeldcode gemaakt met zeven elementen, die allemaal de waarde 0 bevatten:

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

Tot slot kunt u met de constructor `Vector.<T>()` ook een vector met een vaste lengte maken door `true` als tweede parameter (de parameter `fixed`) door te geven. In dat geval wordt de vector met het opgegeven aantal elementen gemaakt en kan het aantal elementen niet worden gewijzigd. Let wel: in een vector met vaste lengte kunt u echter wel de waarden van de elementen wijzigen.

De constructor voor de letterlijke Vector-syntaxis gebruiken

In Flash Professional CS5 of hoger, Flash Builder 4 of hoger en Flex 4 of hoger kunt u een lijst met waarden aan de `Vector.<T>()`-constructor doorgeven om de beginwaarden van de vector op te geven:

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

De volgende informatie is op deze syntaxis van toepassing:

- De navolgende komma is optioneel.
- Er is geen ondersteuning voor lege items in de array. Een instructie als `var v:Vector.<int> = new <int>[0,,2,]` genereert een compilatiefout.
- U kunt geen standaardlengte voor de Vector-instantie opgeven. De lengte is in dit geval identiek aan het aantal elementen in de initialisatielijst.
- U kunt niet opgeven of de Vector-instantie een vaste lengte heeft. Gebruik in dit geval de eigenschap `fixed`.
- Gegevensverlies of fouten kunnen zich voordoen als de items die als waarden worden doorgegeven, niet overeenkomen met het opgegeven type. Bijvoorbeeld:

  ```
  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
  ```

De globale functieVector.<T>() gebruiken

Naast de constructor `Vector.<T>()` en de constructors voor de letterlijke Vector-syntaxis kunt u ook de globale functie `Vector.<T>()` gebruiken om een Vector-object te maken. De `Vector.<T>()` is een conversiefunctie. Als u de globale functie `Vector.<T>()` oproept, geeft u het basistype op van de vector die de methode retourneert. U geeft een enkele geïndexeerde array (Array- of Vector-instantie) als argument door. De methode retourneert vervolgens een vector van het opgegeven basistype, die de waarden in het argument voor de bronarray bevat. In de volgende voorbeeldcode wordt geïllustreerd hoe u de globale functie `Vector.<T>()` globale functie:

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

De `Vector.<T>()` voert gegevenstypeconversie uit op twee niveaus. Wanneer een Array-instantie aan de functie wordt doorgegeven, wordt eerst een Vector-instantie geretourneerd. Afhankelijk van het feit of de bronarray een Array- of een Vector-instantie is, probeert de functie vervolgens de elementen van de bronarray te converteren naar waarden van het basistype. Voor de conversie worden de standaard conversieregels voor ActionScript-gegevenstypen gehanteerd. In de volgende voorbeeldcode worden String-waarden in de bronarray geconverteerd naar gehele getallen in de resulterende vector. Het decimale gedeelte van de eerste waarde (`'1.5'`) wordt afgekapt, en de niet-numerieke derde waarde (`'Waffles'`) is in het resultaat geconverteerd naar 0:

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

Als bepaalde bronelementen niet geconverteerd kunnen worden, treedt een fout op.

Als de code de globale functie `Vector.<T>()` oproept en een element in de bronarray een instantie is van een subklasse van het opgegeven basistype, wordt het element toegevoegd aan de resulterende vector (er treden geen fouten op). Het gebruik van de globale functie `Vector.<T>()` is de enige manier om een vector met basistype `T` te converteren naar een vector met een basistype dat een bovenliggende klasse is van `T`.

Arrayelementen invoegen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De meest fundamentele manier om een element aan een geïndexeerde array toe te voegen, is middels de operator voor arraytoegang (`[]`). Als u de waarde van een geïndexeerd arrayelement wilt instellen, gebruikt u de naam en het indexnummer van het Array- of Vector-object aan de linkerzijde van een toewijzingsinstructie:

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

Als de array of vector nog geen element op die index heeft, wordt de index gemaakt en wordt de waarde daar opgeslagen. Als wel een waarde op die index bestaat, wordt de bestaande waarde door de nieuwe vervangen.

Met een Array-object kunt u een element op een willekeurige index maken. Met een Vector-object kunt u echter alleen waarden toewijzen aan een bestaande index of aan de volgende beschikbare index. De volgende beschikbare index komt overeen met de eigenschap `length` van het Vector-object. De veiligste wijze om een nieuw element aan een Vector-object toe te wijzen, is zoals in deze voorbeeldcode wordt geïllustreerd:

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

Met drie van de methoden voor de klasse Array en Vector - `push()`, `unshift()` en `splice()` - kunt u elementen in een geïndexeerde array invoegen. De methode `push()` voegt een of meer elementen toe aan het einde van een array. Het element dat als laatste met de methode `push()` wordt toegevoegd, krijgt dus het hoogste indexnummer. De methode `unshift()` voegt een of meer elementen toe aan het begin van een array, altijd bij indexnummer 0. De methode `splice()` voegt het gewenste aantal items toe bij de opgegeven index in de array.

In het volgende voorbeeld wordt het gebruik van alle drie de methoden getoond. Er wordt een array met de naam `planets` gemaakt waarin de namen van de planeten worden opgeslagen in volgorde van hun nabijheid bij de zon. Ten eerste wordt de methode `push()` aangeroepen om het eerste item, `Mars` toe te voegen. Vervolgens wordt de methode `unshift()` aangeroepen om het item toe te voegen dat aan het begin van de array hoort, `Mercury`. Ten slotte wordt de methode `splice()` aangeroepen om de items `Venus` en `Earth` toe te voegen achter `Mercury`, maar vóór `Mars`. Het eerste argument dat aan `splice()` wordt doorgegeven, het gehele getal 1, geeft aan dat het invoegen moet beginnen bij index 1. Het tweede argument voor `splice()`, het gehele getal 0, geeft aan dat geen enkel item mag worden verwijderd. Het derde en vierde argument die aan `splice()` worden doorgegeven, `Venus` en `Earth`, zijn de items die moeten worden ingevoegd.

```
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
```

Met zowel de methode `push()` als de methode `unshift()` wordt een geheel getal zonder teken geretourneerd, dat de lengte van de gewijzigde array aangeeft. Wanneer de methode `splice()` wordt gebruikt om elementen in te voegen, wordt een lege array geretourneerd. Dit lijkt misschien vreemd, maar in het licht van de veelzijdigheid van de methode `splice()` klopt het wel. U kunt de methode `splice()` immers niet alleen gebruiken om elementen aan een array toe te voegen, maar ook om elementen uit een array te verwijderen. Wanneer elementen worden verwijderd met de methode `splice()`, wordt een array met daarin de verwijderde elementen geretourneerd.

Opmerking: *Als de eigenschap `fixed` van een Vector-object `true` is, kan het totale aantal elementen in de vector niet veranderen. Als u met een van de hier beschreven technieken een nieuw element probeert toe te voegen aan een vector met vaste lengte, treedt een fout op.*

Waarden ophalen en arrayelementen verwijderen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De meest fundamentele manier om de waarde van een element uit een geïndexeerde array op te halen, is middels de operator voor arraytoegang (`[]`). Als u de waarde van een element uit een geïndexeerde array wilt ophalen, gebruikt u de naam en het indexnummer van het Array- of Vector-object aan de rechterzijde van een toewijzingsinstructie:

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

U kunt proberen een waarde uit een array of vector op te halen met een index waarop geen element bestaat. In dat geval retourneert een Array-object de waarde 'ongedefinieerd' en wordt een RangeError-uitzondering gegenereerd.

Met drie methoden van de klasse Array en Vector - `pop()`, `shift()` en `splice()` - kunt u elementen verwijderen. De methode `pop()` verwijdert een element aan het einde van een array. Het element met het hoogste indexnummer wordt dus verwijderd. De methode `shift()` verwijdert een element aan het begin van de array. Dit betekent dat altijd het element bij indexnummer 0 wordt verwijderd. De methode `splice()`, die ook kan worden gebruikt voor het invoegen van elementen, verwijdert een willekeurig aantal elementen, te beginnen bij het indexnummer dat in het eerste argument aan de methode wordt doorgegeven.

In het volgende voorbeeld wordt het gebruik van de drie methoden getoond om elementen uit een Array-instantie te verwijderen. Er wordt een array met de naam `oceans` gemaakt waarin de namen van de oceanen worden opgeslagen. Sommige namen in de array zijn een meer in plaats van een oceaan en moeten dus worden verwijderd.

Eerst wordt de methode `splice()` gebruikt om de items `Aral` en `Superior` te verwijderen en de items `Atlantic` en `Indian` in te voegen. Het eerste argument dat aan `splice()` wordt doorgegeven (het gehele getal 2), geeft aan dat de bewerking moet beginnen bij het derde item van de lijst (bij index 2). Het tweede argument (2) geeft aan dat twee items moeten worden verwijderd. De overige argumenten (`Atlantic` en `Indian`) zijn de waarden die bij index 2 moeten worden ingevoegd.

Vervolgens wordt de methode `pop()` gebruikt om het laatste element van de array (`Huron`) te verwijderen. Ten slotte wordt met de methode `shift()` het eerste item van de array (`Victoria`) verwijderd.

```
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
```

Zowel de methode `pop()` als de methode `shift()` retourneert het item dat is verwijderd. Voor een Array-instantie is het gegevenstype van de resultaatwaarde Object omdat arrays waarden van elk willekeurig gegevenstype kunnen bevatten. Voor een Vector-instantie is het gegevenstype van de resultaatwaarde het basistype van de vector. De methode `splice()` retourneert een array of vector met daarin de verwijderde elementen. U kunt het Array-voorbeeld met `oceans` bijvoorbeeld zo wijzigen dat de geretourneerde array door het oproepen van `splice()` wordt toegewezen aan een nieuwe Array-variabele, zoals in het volgende voorbeeld wordt getoond:

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

U kunt code tegenkomen waarin de operator `delete` wordt gebruikt voor een Array-objectelement. Met de operator `delete` wordt de waarde van een Array-element op `undefined` ingesteld, maar wordt het element niet uit de array verwijderd. De volgende code gebruikt de operator `delete` bijvoorbeeld voor het derde element in de array `oceans`, maar de lengte van de array blijft 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
```

U kunt een array of vector afkappen door de eigenschap `length` van de array te gebruiken. Als u de eigenschap `length` van een geïndexeerde array instelt op een lengte die kleiner is dan de huidige lengte van de array, wordt de array afgekapt en worden alle elementen verwijderd die zijn opgeslagen op indexnummers hoger dan de nieuwe waarde van `length` min 1. Als de array `oceans` bijvoorbeeld zo was gesorteerd dat alle geldige vermeldingen zich aan het begin van de array bevonden, zou u de eigenschap `length` kunnen gebruiken om de vermeldingen aan het einde van de array te verwijderen, zoals in de volgende voorbeeldcode wordt geïllustreerd:

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

Opmerking: *Als de eigenschap `fixed` van een Vector-object `true` is, kan het totale aantal elementen in de vector niet veranderen. Als u met een van de hier beschreven technieken een element probeert te verwijderen uit een vector met vaste lengte of een vector met vaste lengte probeert af te kappen, treedt een fout op.*

Een array sorteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er bestaan drie methoden - `reverse()`, `sort()` en `sortOn()` - om de volgorde van een geïndexeerde array te wijzigen, ofwel door te sorteren ofwel door de volgorde om te keren. Al deze methoden wijzigen de bestaande array. De volgende tabel bevat een overzicht van deze methoden en hun gedrag voor Array- en Vector-objecten:

Methode	Arraygedrag	Vectorgedrag
`reverse()`	Deze methode wijzigt de volgorde van de array zo dat het laatste element het eerste wordt. Het op één na laatste element wordt het tweede element enzovoort.	Identiek aan arraygedrag
`sort()`	Met deze methode kunt u de elementen in de array op diverse, vooraf gedefinieerde manieren sorteren, bijvoorbeeld in alfabetische of numerieke volgorde. U kunt ook een aangepaste sorteeralgoritme opgeven.	Deze methode sorteert elementen volgens de aangepaste sorteeralgoritme die u opgeeft.
`sortOn()`	Met deze methode kunt u objecten sorteren die een of meer gemeenschappelijke eigenschappen hebben, door de eigenschap(pen) als sorteersleutel op te geven.	Niet beschikbaar in de klasse Vector

De methode reverse()

De methode `reverse()` gebruikt geen parameters en retourneert geen waarde, maar maakt het mogelijk om de volgorde van de array om te wisselen van de huidige toestand naar de omgekeerde volgorde. In het volgende voorbeeld wordt de volgorde van de oceanen in de array `oceans` omgekeerd:

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

Basissortering met de methode sort() (alleen klasse Array)

Voor een Array-instantie rangschikt de methode `sort()` de elementen van de array opnieuw in de *standaard sorteervolgorde*. De standaardvolgorde heeft de volgende kenmerken:

- Bij het sorteren wordt onderscheid gemaakt tussen hoofdletters en kleine letters (hoofdletters komen voor de kleine letters). De letter D komt dus voor de letter b.
- Er wordt oplopend gesorteerd. Een lagere tekencode (zoals de A) komt voor de hogere tekencodes (zoals de B).
- Identieke waarden worden bij het sorteren naast elkaar geplaatst, maar niet in een bepaalde volgorde.
- Voor het sorteren worden tekenreeksen gebruikt. Dit wil zeggen dat elementen eerst worden omgezet in een tekenreeks, voordat ze met elkaar worden vergeleken (dus 10 komt voor 3 omdat de tekenreeks `"1"` een lagere tekencode heeft dan de tekenreeks `"3"`).

Misschien wilt u een array sorteren zonder rekening te houden met de hoofdletters en kleine letters, of in aflopende volgorde, of misschien bevat uw array getallen die u numeriek wilt sorteren in plaats van alfabetisch. Bij de methode `sort()` van de klasse Array hoort de parameter `options`, waarmee u de verschillende kenmerken van de standaardvolgorde kunt aanpassen. De opties worden gedefinieerd met een set statische constanten in de klasse Array, zoals wordt getoond in de volgende lijst:

- `Array.CASEINSENSITIVE`: hiermee wordt bij het sorteren niet gekeken naar de hoofdletters en kleine letters. De kleine letter b komt dus voor de hoofdletter D.
- `Array.DESCENDING`: hiermee wordt de standaardvolgorde waarmee wordt gesorteerd, omgekeerd. De letter B komt dus voor de letter A.

- `Array.UNIQUESORT`: hiermee wordt het sorteren afgebroken als twee identieke waarden worden aangetroffen.
- `Array.NUMERIC`: hiermee wordt numeriek gesorteerd, dus 3 komt voor 10.

In het volgende voorbeeld worden enkele van deze opties toegepast. Er wordt een array met de naam `poets` gemaakt, die vervolgens wordt gesorteerd met behulp van verschillende opties.

```
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
```

Aangepaste sortering met de methode sort() (klassen Array en Vector)

Naast de basissortering die voor een Array-object beschikbaar is, kunt u ook een aangepaste sorteerregel definiëren. Deze techniek is de enige vorm van de methode `sort()` die beschikbaar is voor de klasse Vector. Als u een aangepaste sortering wilt opgeven, schrijft u een aangepaste sorteerfunctie en geeft u deze als argument door aan de methode `sort()`.

Stel dat u een lijst met namen hebt waarbij elk element van de lijst bestaat uit de volledige naam van een persoon, maar u wilt de lijst sorteren op achternaam. In dit geval moet u een aangepaste sorteerfunctie gebruiken om de elementen te parseren en de achternaam te gebruiken in de sorteerfunctie. In de volgende code wordt getoond hoe u dit kunt doen met een aangepaste functie die wordt gebruikt als parameter voor de methode `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
```

De aangepaste sorteerfunctie `orderLastName()` gebruikt een reguliere expressie om uit elk element de achternaam te extraheren voor de vergelijking. De functie-id `orderLastName` wordt als enige parameter gebruikt bij het aanroepen van de methode `sort()` voor de array `names`. De sorteerfunctie accepteert twee parameters (`a` en `b`), omdat de functie met twee arrayelementen tegelijk aan het werk is. De geretourneerde waarde van de sorteerfunctie geeft aan hoe de elementen moeten worden gesorteerd:

- Wanneer -1 wordt geretourneerd, wil dit zeggen dat de eerste parameter (`a`) voor de tweede parameter (`b`) komt.
- Wanneer 1 wordt geretourneerd, wil dit zeggen dat de tweede parameter (`b`) voor de eerste parameter (`a`) komt.
- Wanneer 0 wordt geretourneerd, wil dit zeggen dat de elementen dezelfde sorteerprioriteit hebben.

De methode sortOn() (alleen klasse Array)

De methode `sortOn()` is bedoeld voor Array-objecten met elementen die objecten bevatten. Deze objecten worden geacht ten minste één gemeenschappelijke eigenschap te hebben voor gebruik als sorteersleutel. Het gebruik van de methode `sortOn()` voor arrays van een ander type geeft onverwachte resultaten.

Opmerking: *De klasse Vector kent geen methode `sortOn()`. Deze methode is alleen beschikbaar voor Array-objecten.*

In het volgende voorbeeld wordt de array `poets` zo aangepast dat elk element een object is in plaats van een tekenreeks. Elk object bevat de achternaam en het geboortejaar van de dichter.

```
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"});
```

U kunt de methode `sortOn()` gebruiken om de array te sorteren op de eigenschap `born`. De methode `sortOn()` definieert twee parameters: `fieldName` en `options`. Het argument `fieldName` moet worden opgegeven als tekenreeks. In het volgende voorbeeld wordt `sortOn()` aangeroepen met twee argumenten ("`born`" en `Array.NUMERIC`). Het argument `Array.NUMERIC` dient om ervoor te zorgen dat numeriek in plaats van alfabetisch wordt gesorteerd. Dit is een goede manier van werken, zelfs wanneer alle getallen uit hetzelfde aantal cijfers bestaan, omdat de sorteeropdracht altijd zoals verwacht werkt als later een getal met minder of meer cijfers aan de array wordt toegevoegd.

```
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
*/
```

Sorteren zonder de oorspronkelijke array te wijzigen (alleen klasse Array)

Over het algemeen wordt met de methoden `sort()` en `sortOn()`= de array gewijzigd. Als u een array wilt sorteren zonder de bestaande array te wijzigen, moet u de constante `Array.RETURNINDEXEDARRAY` doorgeven als onderdeel van de parameter `options`. Deze optie geeft de methoden de opdracht om een nieuwe array met de gewenste sorteervolgorde te retourneren en om de oorspronkelijke array ongewijzigd te laten. De array die door deze methoden wordt geretourneerd, is een eenvoudige array met indexnummers die de nieuwe sorteervolgorde in acht neemt en geen elementen van de oorspronkelijke array bevat. Als u bijvoorbeeld de array `poets` op geboortejaar wilt sorteren zonder de array te wijzigen, voegt u de constante `Array.RETURNINDEXEDARRAY` toe als onderdeel van het argument dat wordt doorgegeven voor de parameter `options`.

In het volgende voorbeeld wordt de geretourneerde indexinformatie opgeslagen in de array `indices` en wordt de array `indices` samen met de ongewijzigde array `poets` gebruikt om de dichters in volgorde van geboortejaar te krijgen:

```
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
*/
```

Zoeken in een array

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Vier methoden van de klassen Array en Vector - `concat()`, `join()`, `slice()` en `toString()` - dienen allemaal om in een array te zoeken naar informatie. De array wordt niet gewijzigd. Met de methoden `concat()` en `slice()` wordt een nieuwe array geretourneerd, terwijl zowel de methode `join()` als de methode `toString()` een tekenreeks oplevert. De methode `concat()` neemt een nieuwe array of lijst met elementen als argumenten en maakt hier in combinatie met de bestaande array een nieuwe array van. De methode `slice()` heeft twee parameters met de toepasselijke naam `startIndex` en `endIndex` en retourneert een nieuwe array met daarin een kopie van de elementen uit de bestaande array. Dit begint bij het element dat zich bevindt bij `startIndex` en eindigt bij het element net voor `endIndex`. Dus: het element bij `endIndex` zit niet in de geretourneerde waarde.

In het volgende voorbeeld worden `concat()` en `slice()` gebruikt om nieuwe arrays te maken met elementen uit een andere array:

```
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
```

Met de methode `join()` en `toString()` kunt u zoeken in de array en de inhoud ervan retourneren als tekenreeks. Als geen parameters worden gebruikt voor de methode `join()`, gedragen beide methoden zich op dezelfde manier: ze retourneren een tekenreeks bestaande uit een door komma's gescheiden lijst van alle elementen in de array. In tegenstelling tot de methode `toString()` accepteert de methode `join()` een parameter die `delimiter` heet. Met deze parameter kunt u het symbool kiezen dat u wilt gebruiken als scheidingsteken tussen de verschillende elementen in de geretourneerde tekenreeks.

In het volgende voorbeeld wordt de array `rivers` gemaakt. Hierbij wordt zowel `join()` als `toString()` opgeroepen om de waarden in de array te retourneren als een tekenreeks. De methode `toString()` wordt gebruikt om door komma's gescheiden waarden te retourneren (`riverCSV`), terwijl met de methode `join()` waarden worden geretourneerd die door het teken + van elkaar worden gescheiden.

```
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
```

Iets om rekening mee te houden bij de methode `join()` is dat geneste Array- of Vector-instanties altijd worden geretourneerd met waarden die met een komma van elkaar gescheiden zijn, ongeacht het scheidingsteken dat u hebt opgegeven voor de elementen van de hoofdarray. Dit wordt getoond in het volgende voorbeeld:

```
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
```

Associatieve arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een associatieve array, die ook wel een *hash* of *map* wordt genoemd, gebruikt *sleutels* in plaats van een numerieke index om opgeslagen waarden te ordenen. Elke sleutel in een associatieve array is een unieke tekenreeks die toegang geeft tot een opgeslagen waarde. Een associatieve array is een instantie van de klasse Object, wat wil zeggen dat elke sleutel overeenkomt met een eigenschapnaam. Associatieve arrays zijn ongeordende verzamelingen van sleutel- en waardeparen. In de code wordt er niet vanuit gegaan dat de sleutels van een associatieve array in een specifieke volgorde staan.

ActionScript 3.0 bevat eveneens een geavanceerd type van associatieve arrays, die *woordenboeken* worden genoemd. Woordenboeken, instanties van de klasse Dictionary in het pakket flash.utils, werken met sleutels die elk willekeurig gegevenstype kunnen hebben. Woordenboeksleutels zijn met andere woorden niet beperkt tot waarden van het type String.

Associatieve arrays met tekenreekssleutels

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn twee manieren om associatieve arrays te maken in ActionScript 3.0. De eerste manier is met behulp van een Object-instantie. Met een Object-instantie kunt u de array initialiseren met een objectliteraal. Een instantie van de klasse Object, ook een *algemeen object* genoemd, is functioneel gezien identiek aan een associatieve array. Elke eigenschapnaam van het algemene object wordt gebruikt als de sleutel die toegang geeft tot een opgeslagen waarde.

In het volgende voorbeeld wordt een associatieve array gemaakt met de naam `monitorInfo` en daarbij wordt een letterlijk teken voor objecten gebruikt om de array te initialiseren met twee sleutel- en waardeparen.

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

Als u de array niet hoeft te declareren op het declaratietijdstip, kunt u de constructor Object gebruiken om de array te maken. Dit gaat als volgt:

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

Nadat de array is gemaakt met een objectliteraal of de constructor van de klasse Object, kunt u nieuwe waarden aan de array toevoegen met de operator voor arraytoegang (`[]`) of de puntoperator (`.`). In het volgende voorbeeld worden twee nieuwe waarden toegevoegd aan `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
```

De sleutel met de naam `aspect ratio` bevat een spatieteken. Dit is mogelijk wanneer u de operator voor arraytoegang (`[]`) gebruikt. Er wordt echter een fout gegenereerd als u dit probeert met de puntoperator. Het gebruik van spaties in sleutelnamen wordt afgeraden.

De tweede manier om een associatieve array te maken is de Array-constructor (of de constructor van een willekeurige dynamische klasse) te gebruiken en vervolgens de operator voor arraytoegang (`[]`) of de puntoperator (`.`) te gebruiken om sleutel- en waardeparen aan de array toe te voegen. Als u uw associatieve array als type Array declareert, kunt u geen letterlijk teken voor objecten gebruiken om de array te initialiseren. In het volgende voorbeeld wordt een associatieve array met de naam `monitorInfo` gemaakt met de constructor Array en worden een sleutel met de naam `type` en een sleutel met de naam `resolution` toegevoegd, samen met hun waarden:

```
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
```

Het biedt geen voordelen de constructor Array te gebruiken om een associatieve array te maken. U kunt de eigenschap `Array.length` of de methoden uit de klasse Array niet gebruiken met associatieve arrays, zelfs bij gebruik van de constructor Array of het gegevenstype Array. De constructor Array is het meest geschikt voor het maken van geïndexeerde arrays.

Geassocieerde arrays met objectsleutels (Woordenboeken)

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse Dictionary kunt u een associatieve array maken die objecten als sleutel gebruikt in plaats van tekenreeksen. Dergelijke arrays worden soms dictionary's, hashes of maps genoemd. Neem bijvoorbeeld een toepassing die de locatie van een object Sprite bepaalt op basis van zijn associatie met een specifieke container. Met een object Dictionary kunt u elk object Sprite aan een container koppelen.

Met de volgende code worden drie instanties van de klasse Sprite gemaakt die dienen als sleutel voor het object Dictionary. Elke sleutel krijgt een waarde toegewezen uit `GroupA` of `GroupB`. De waarden kunnen van elk gegevenstype zijn, maar in dit voorbeeld is zowel `GroupA` als `GroupB` een instantie uit de klasse Object. U kunt de waarde die aan de sleutels is gekoppeld, benaderen met de operator voor arraytoegang (`[]`), zoals in de volgende code wordt getoond:

```
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");
}
```

Doorlopen met objectsleutels

U kunt de inhoud van een object Dictionary doorlopen met een lus `for..in` of een lus `for each..in` de lus. Met een lus `for..in` kunt u de inhoud doorlopen op basis van de sleutels en met een lus `for each..in` kunt u de inhoud doorlopen op basis van de waarden die aan elke sleutel zijn gekoppeld.

Gebruik de lus `for..in` voor directe toegang tot de objectsleutels van een object Dictionary. U kunt ook de waarden van het Dictionary-object benaderen met de operator voor arraytoegang (`[]`). In de volgende code wordt het vorige voorbeeld van de dictionary `groupMap` gebruikt om aan te geven hoe u een object Dictionary doorloopt met de lus `for..in` de lus :

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

Gebruik de lus `for each..in` voor directe toegang tot de waarden van een object Dictionary. In de volgende code wordt ook de dictionary `groupMap` gebruikt om aan te geven hoe u een object Dictionary doorloopt met de lus `for each..in` de lus :

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

Objectsleutels en geheugenbeheer

Adobe® Flash® Player en Adobe® AIR™ gebruiken een opschoonsysteem om geheugen vrij te maken dat niet meer in gebruik is. Wanneer niet meer naar een object wordt verwezen, kan het worden opgeschoond. De volgende keer dat de opschoonfunctie wordt uitgevoerd, wordt geheugen vrijgemaakt. In de volgende code wordt bijvoorbeeld een nieuw object gemaakt en wordt een verwijzing naar het object toegewezen aan de variabele `myObject`:

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

Zolang er verwijzingen naar het object bestaan, maakt de opschoonfunctie het geheugen dat het object in beslag neemt, niet vrij. Als de waarde van `myObject` zodanig wordt gewijzigd dat er een verwijzing naar een ander object ontstaat of dat de waarde `null` wordt, komt het geheugen dat door het oorspronkelijke object wordt ingenomen in aanmerking om te worden opgeschoond, maar alleen als er geen andere verwijzingen naar het oorspronkelijke object bestaan.

Als u `myObject` gebruikt als sleutel in een object Dictionary, maakt u nog een verwijzing naar het oorspronkelijke object. In de volgende code worden bijvoorbeeld twee verwijzingen naar een object gemaakt, de variabele `myObject` en de sleutel in het object `myMap`:

```
import flash.utils.Dictionary;

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

Als u het object waarnaar wordt verwezen door `myObject` wilt laten opschonen, moet u alle verwijzingen ernaar verwijderen. In dit geval moet u de waarde `myObject` wijzigen en de sleutel `myObject` verwijderen uit `myMap`, zoals in de volgende code wordt getoond:

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

U kunt echter ook de parameter `useWeakReference` van de constructor Dictionary gebruiken om van alle dictionarysleutels een zogenaamde *zwakke verwijzing* te maken. Zwakke verwijzingen worden door het opschoonsysteem genegeerd, wat betekent dat een object met alleen zwakke verwijzingen voor opschonen in aanmerking komt. In de volgende code hoeft u bijvoorbeeld de sleutel `myObject` niet te verwijderen uit `myMap` om het object te laten opschonen:

```
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.
```

Multidimensionale arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Multidimensionale arrays bevatten andere arrays als elementen. Kijk bijvoorbeeld naar dit voorbeeld van een takenlijst die wordt opgeslagen als een geïndexeerde array van tekenreeksen.

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

Als u een afzonderlijke takenlijst wilt opslaan voor elke dag van de week, kunt u een multidimensionale array maken met één element voor elke dag van de week. Elk element bevat een geïndexeerde array, ongeveer zoals de array `tasks`, die de takenlijst opslaat. In een multidimensionale array kunt u elke combinatie van geïndexeerde of associatieve arrays gebruiken. In de voorbeelden in de volgende secties worden twee geïndexeerde arrays of wordt een associatieve array van geïndexeerde arrays gebruikt. Als oefening kunt u andere combinaties proberen.

Twee geïndexeerde arrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u twee geïndexeerde arrays gebruikt, kunt u het resultaat zichtbaar maken in de vorm van een tabel of spreadsheet. De elementen van de eerste array stellen de rijen van de tabel voor. De elementen van de tweede array zijn voor de kolommen.

De volgende multidimensionale array gebruikt bijvoorbeeld twee geïndexeerde arrays om takenlijsten bij te houden voor de dagen van de week. De eerste array (`masterTaskList`) wordt gemaakt met de klassenconstructor Array. Elk element van de array is een dag van de week. Index 0 is maandag en index 6 is zondag. Deze elementen kunnen worden beschouwd als de rijen van de tabel. U kunt voor elke dag een takenlijst maken door een letterlijk arrayteken toe te wijzen aan elk van de zeven elementen die u maakt in de array `masterTaskList`. De letterlijke arraytekens zijn de kolommen van de tabel.

```
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"];
```

U kunt de afzonderlijke items in een van de taaklijsten benaderen met de operator voor arraytoegang (`[]`). De eerste set haakjes wordt gebruikt voor de dag van de week. De tweede set voor de takenlijst van die dag. Als u bijvoorbeeld de tweede taak van het lijstje van woensdag wilt opvragen, gebruikt u eerst index 2 voor woensdag en vervolgens index 1 voor de tweede taak van de lijst.

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

Als u de eerste taak van het lijstje van zondag wilt opvragen, gebruikt u eerst index 6 voor zondag en vervolgens index 0 voor de eerste taak van de lijst.

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

Associatieve array met een geïndexeerde array

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de afzonderlijke arrays beter toegankelijk maken door een associatieve array te gebruiken voor de dagen van de week en een geïndexeerde array voor de takenlijsten. Met een associatieve array kunt u de puntnotatie gebruiken voor verwijzingen naar een bepaalde dag van de week. Dit gaat wel ten koste van de uitvoeringstijd, omdat elk element van de associatieve array moet worden benaderd. In het volgende voorbeeld wordt een associatieve array gebruikt als basis voor een takenlijst, met een sleutel-waardepaar voor elke dag van de week:

```
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"];
```

De puntnotatie maakt de code beter leesbaar, omdat op deze manier meerdere sets haakjes kunnen worden vermeden.

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

U kunt de takenlijst doorlopen met een lus `for..in lus`, maar u moet de operator voor arraytoegang (`[]`) in plaats van de puntsyntaxis gebruiken om toegang te krijgen tot de waarde die aan elke sleutel is gekoppeld. Omdat `masterTaskList` een associatieve array is, worden de elementen niet noodzakelijkerwijs opgevraagd in de volgorde die u zou verwachten, zoals in het volgende voorbeeld wordt getoond:

```
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
*/
```

Arrays klonen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Array heeft geen ingebouwde methode voor het kopiëren van arrays. U kunt een *oppervlakkigekopie* van een array maken door de methode `concat()` of `slice()` aan te roepen zonder argumenten. Bij een oppervlakkige kopie geldt dat als de originele array elementen heeft die objecten zijn, alleen de verwijzingen naar de objecten worden gekopieerd in plaats van de objecten zelf. De kopie verwijst naar dezelfde objecten als het origineel. Wanneer de objecten worden gewijzigd, worden deze wijzigingen door beide arrays overgenomen.

Bij een *diepe kopie* worden alle objecten die in de originele array worden aangetroffen ook gekopieerd. De nieuwe array verwijst dus niet naar dezelfde objecten als de originele array. Diep kopiëren vergt meer dan één regel code. Meestal moet een functie worden gemaakt. Zo'n functie kan worden gemaakt als algemene hulpprogrammafunctie of als methode van een subklasse Array.

In het volgende voorbeeld wordt een functie met de naam `clone()` gedefinieerd voor een diepe kopie. Het algoritme wordt geleend uit een algemene Java-programmeertechniek. De functie maakt een diepe kopie door de array met serienummering te coderen in een instantie van de klasse ByteArray en de array vervolgens opnieuw in te lezen in een nieuwe array. Deze functie accepteert een object, zodat het kan worden gebruikt met zowel geïndexeerde als associatieve arrays, zoals in de volgende code wordt getoond:

```
import flash.utils.ByteArray;

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

De klasse Array uitbreiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Array is een van de weinige kernklassen die niet definitief is. Dit betekent dat u voor Array uw eigen subklassen kunt maken. In deze sectie wordt een voorbeeld gegeven van de manier waarop u een subklasse voor Array kunt maken. Ook komen enkele problemen aan bod die hierbij kunnen optreden.

Zoals eerder is vermeld, worden arrays in ActionScript niet getypeerd. Het is echter wel mogelijk om een subklasse voor Array te maken die alleen elementen van een specifiek gegevenstype accepteert. In het voorbeeld in de volgende secties wordt voor Array een subklasse met de naam TypedArray gedefinieerd die de elementen beperkt tot waarden van het gegevenstype dat in de eerste parameter is opgegeven. De klasse TypedArray wordt hier slechts gegeven als voorbeeld van hoe de klasse Array kan worden uitgebreid. Deze manier van werken is om diverse redenen mogelijk niet geschikt voor productiewerk. Ten eerste vindt controle van het type plaats tijdens het uitvoeren en niet tijdens het compileren. Ten tweede: wanneer een methode TypedArray gegevens aantreft die niet overeenkomen, wordt dit genegeerd en wordt geen uitzondering gegenereerd, hoewel de methoden gemakkelijk kunnen worden aangepast om wel uitzonderingen te genereren. Ten derde kan de klasse niet voorkomen dat de operator voor arraytoegang wordt gebruikt om waarden van een willekeurig type in te voegen in de array. Ten vierde is deze manier van coderen meer gericht op eenvoud dan optimale prestaties.

Opmerking: *Met de hier beschreven technieken kunt u een array met type maken. Het is echter beter om daarvoor een Vector-object te gebruiken. Een Vector-instantie is een echte array met type, en biedt betere prestaties en andere voordelen ten opzichte van de klasse Array of een subklasse. Het doel van deze discussie is aan te tonen hoe een subklasse van de klasse Array wordt gemaakt.*

De subklasse declareren

Gebruik het trefwoord `extends` om aan te geven dat een klasse een subklasse van Array is. Een subklasse van Array moet het kenmerk `dynamic` gebruiken (net als de klasse Array). Anders werkt de subklasse niet goed.

De volgende code toont hoe de klasse TypedArray wordt gedefinieerd, met daarin een constante voor het gegevenstype, een constructormethode en de vier methoden waarmee elementen aan de array kunnen worden toegevoegd. In dit voorbeeld wordt de code voor de verschillende methoden weggelaten. In latere secties wordt een en ander wel volledig uitgelegd:

```
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 {}
}
```

De vier overschreven methoden gebruiken allemaal de naamruimte AS3 in plaats van het kenmerk `public`, omdat er in dit voorbeeld van wordt uitgegaan dat de compileroptie `-as3` op `true` en `-es` op `false` staat. Dit zijn standaardinstellingen voor Adobe Flash Builder en voor AdobeFlashProfessional.

Als u een gevorderd ontwikkelaar bent die liever werkt met prototypeovererving, kunt u de klasse TypedArray op twee kleine punten aanpassen, zodat wordt gecompileerd met de compileroptie `-es` op `true`. Eerst verwijdert u het kenmerk `override` overal waar het voorkomt en vervangt u de naamruimte AS3 door het kenmerk `public`. Vervolgens gebruikt u `Array.prototype` voor alle vier de keren dat `super` voorkomt.

De constructor TypedArray

De subklassenconstructor vormt een interessante uitdaging, omdat de constructor een lijst met argumenten van willekeurige lengte moet accepteren. De uitdaging zit hem in de manier waarop de argumenten worden doorgegeven aan de superconstructor om de array te maken. Als u de lijst met argumenten als een array doorgeeft, wordt deze gezien als één argument van het type Array en is de resulterende array altijd 1 element lang. De traditionele manier om argumentenlijsten door te geven is met de methode `Function.apply()`. In deze methode vormt een array met argumenten een tweede parameter die wordt geconverteerd naar een lijst met argumenten wanneer de functie wordt uitgevoerd. Helaas kan de methode `Function.apply()` niet worden gebruikt bij constructors.

De enige resterende mogelijkheid is dat de logica van de constructor Array wordt nagemaakt in de constructor TypedArray. De volgende code laat de algoritme zien die in de klassenconstructor Array wordt gebruikt en die u ook kunt gebruiken in uw constructor voor een subklasse voor 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]
            }
        }
    }
}
```

De constructor TypedArray heeft de meeste code gemeenschappelijk met de constructor Array. Er zijn slechts op vier punten verschillen. Ten eerste bevat de parameterlijst een nieuwe verplichte parameter van het type Class waarmee het gegevenstype van de arrays kan worden opgegeven. Vervolgens wordt het gegevenstype dat aan de constructor wordt doorgegeven toegewezen aan de variabele `dataType`. Daarna wordt in de instructie `else` de waarde van de eigenschap `length` toegewezen na de lus `for`, zodat `length` alleen argumenten van het juiste type bevat. Ten slotte wordt in de body van de lus `for` de overschreven versie van de methode `push()` gebruikt, zodat alleen argumenten van het juiste gegevenstype aan de array worden toegevoegd. Het volgende voorbeeld laat de constructorfunctie TypedArray zien:

```
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;
        }
    }
}
```

Door TypedArray overschreven methoden

De klasse TypedArray overschrijft de vier methoden van de klasse Array die in staat zijn om elementen toe te voegen aan een array. In elk geval voegt de overschreven methode een typecontrole toe waarmee het toevoegen van elementen van het verkeerde gegevenstype wordt voorkomen. Vervolgens roept elke methode de superklassenversie van zichzelf aan.

De methode `push()` doorloopt de lijst met argumenten met een lus `for..in`, waarbij voor elk argument een typecontrole wordt uitgevoerd. Alle argumenten die niet van het juiste type zijn, worden uit de array `args` verwijderd met de methode `splice()`. Wanneer de lus `for..in` is voltooid, bevat de array `args` alleen waarden van het type `dataType`. Vervolgens wordt de superklassenversie van `push()` aangeroepen met de bijgewerkte array `args`, zoals in de volgende code:

```
    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));
    }
```

De methode `concat()` maakt een tijdelijke TypedArray met de naam `passArgs` om de argumenten op te slaan die de typecontrole doorstaan. Dit maakt het mogelijk om de typecontrolecode die in de methode `push()` aanwezig is opnieuw te gebruiken. Met een lus `for..in` wordt de array `args` doorlopen en wordt `push()` voor elk argument aangeroepen. Omdat `passArgs` wordt getypeerd als TypedArray, wordt de TypedArray-versie van `push()` uitgevoerd. De methode `concat()` roept vervolgens zijn eigen superklassenversie aan, zoals in de volgende code:

```
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));
}
```

De methode `splice()` neemt een willekeurige lijst met argumenten, maar de eerste twee argumenten verwijzen altijd naar een indexnummer en het aantal elementen dat moet worden verwijderd. Daarom voert de overschreven methode `splice()` alleen een controle van het type uit voor elementen in de array `args` op indexpositie 2 of hoger. Interessant aan de code is dat er een recursieve aanroep van `splice()` lijkt te zijn in de lus `for`. Dit is echter geen recursieve aanroep, omdat `args` van het type Array is in plaats van TypedArray. Dit betekent dat het aanroepen van `args.splice()` een aanroep is naar de superklassenversie van de methode. Wanneer de lus `for..in` is voltooid, bevat de array `args` alleen waarden van het juiste type op indexpositie 2 of hoger en roept `splice()` zijn eigen superklassenversie aan, zoals in de volgende code wordt getoond:

```
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));
}
```

De methode `unshift()`, waarmee elementen worden toegevoegd aan het begin van een array, accepteert ook een willekeurige lijst met argumenten. De overschreven methode `unshift()` gebruikt een algoritme die veel lijkt op het algoritme dat door de methode `push()` wordt gebruikt, zoals in het volgende codevoorbeeld wordt getoond:

```
    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));
    }
}
```

Voorbeeld van arrays: PlayList

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het voorbeeld PlayList worden technieken voor het werken met arrays gedemonstreerd in de context van een muziekspeler waarin een lijst met nummers wordt beheerd. Deze technieken zijn:

- Een geïndexeerde array maken
- Items toevoegen aan een geïndexeerde array
- Een array van objecten sorteren op verschillende eigenschappen met behulp van verschillende sorteeropties
- Een array omzetten in een tekenreeks met een teken als scheidingsteken

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de bestanden voor de PlayList-toepassing in de map Samples/PlayList. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
PlayList.mxml of PlayList.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/playlist/PlayList.as	Een klasse die een lijst met nummers voorstelt. Deze gebruikt een Array om de lijst op te slaan, en beheert de sortering van de items in de lijst.
com/example/programmingas3/playlist/Song.as	Een waardeobject voor informatie over een nummer. De items die door de klasse PlayList worden beheerd, zijn instanties van Song.
com/example/programmingas3/playlist/SortProperty.as	Een pseudo-opsomming waarvan de beschikbare waarden de eigenschappen van de klasse Song voorstellen aan de hand waarvan een lijst met objecten van het type Song kan worden gesorteerd.

Overzicht van de klasse PlayList

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse PlayList beheert een set objecten van het type Song. Deze klasse heeft methoden van het type public met functionaliteit voor het toevoegen van een nummer aan de afspeellijst (de methode `addSong()`) en voor het sorteren van de nummers in de lijst (de methode `sortList()`). Bovendien bevat deze klasse een eigenschap voor alleen-lezen-toegang (`songList`) voor toegang tot de eigenlijke set nummers in de afspeellijst. Intern houdt de klasse PlayList de nummers bij met behulp van een Array-variabele van het type private:

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

Naast de Array-variabele `_songs`, die door de klasse PlayList wordt gebruikt om de eigen lijst met nummers bij te houden, houden twee andere private variabelen bij of de lijst moet worden gesorteerd (`_needToSort`) en aan de hand van welke eigenschap de lijst met nummers moet worden gesorteerd op een bepaald moment (`_currentSort`).

Zoals bij alle objecten is het declareren van een instantie Array nog maar de helft van het maken van een array. Voordat de eigenschappen of methoden van een instantie Array kunnen worden benaderd, moet van de array een instantie worden gemaakt met behulp van de constructor van de klasse PlayList.

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

De eerste regel van de constructor maakt een instantie van de variabele `_songs`, zodat deze klaar is voor gebruik. Vervolgens wordt de methode `sortList()` aangeroepen om de eerste eigenschap waarop wordt gesorteerd te bepalen.

Een nummer aan de lijst toevoegen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer een gebruiker een nieuw nummer aan de toepassing toevoegt, roept de code in het gegevensinvoerformulier de methode `addSong()` van de klasse PlayList aan.

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

In `addSong()` wordt de methode `push()` van de array `_songs` aangeroepen, waardoor het object Song dat aan `addSong()` is doorgegeven, als nieuw element aan die array wordt toegevoegd. Met de methode `push()` wordt het nieuwe element aan het einde van de array toegevoegd, ongeacht de eventueel eerder toegepaste sorteervolgorde. Dit betekent dat na het aanroepen van de methode `push()` de sorteervolgorde van de lijst met nummers waarschijnlijk niet meer klopt. Daarom wordt de variabele `_needToSort` op `true` ingesteld. In theorie kan onmiddellijk de methode `sortList()` worden aangeroepen, zodat niet meer hoeft te worden bijgehouden of de lijst op een bepaald moment al dan niet gesorteerd is. In de praktijk hoeft de lijst met nummers echter pas net vóór het opvragen te worden gesorteerd. Door het sorteren uit te stellen hoeft de toepassing niet onnodig te sorteren als bijvoorbeeld meerdere nummers aan de lijst worden toegevoegd vóór het opvragen.

De lijst met nummers sorteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat de instanties Song die door de afspeellijst worden beheerd complexe objecten zijn, willen de gebruikers van de toepassing de afspeellijst misschien liever sorteren op andere eigenschappen, zoals de naam van het nummer of het jaar waarin het is uitgebracht. In de toepassing PlayList bestaat het sorteren van de lijst met nummers uit drie onderdelen: de eigenschap identificeren waarop de lijst moet worden gesorteerd, aangeven welke sorteeropties gebruikt moeten worden wanneer er op die eigenschap wordt gesorteerd en het daadwerkelijk uitvoeren van de sorteerbewerking.

Eigenschappen om op te sorteren.
Een object van het type Song houdt verschillende eigenschappen bij, waaronder de titel van het nummer, de artiest, het jaar waarin het nummer is uitgebracht, de bestandsnaam en een door de gebruiker geselecteerde verzameling genres waartoe het nummer behoort. Hiervan zijn alleen de eerste drie handig om op te sorteren. Om het de ontwikkelaars gemakkelijker te maken wordt in het voorbeeld de klasse SortProperty gebruikt, die fungeert als opsomming met waarden voor de eigenschappen waarop kan worden gesorteerd.

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

De klasse SortProperty bevat drie constanten (`TITLE`, `ARTIST` en `YEAR`). In elk van deze constanten wordt een tekenreeks opgeslagen met de werkelijke naam van de bijbehorende eigenschap van de klasse Song waarop kan worden gesorteerd. In de rest van de code worden sorteereigenschappen aangegeven met het lid van de opsomming. In de constructor PlayList wordt bijvoorbeeld de lijst in eerste instantie gesorteerd door de methode `sortList()` aan te roepen:

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

Omdat de eigenschap om op te sorteren wordt aangegeven als `SortProperty.TITLE`, worden de nummers gesorteerd op titel.

Sorteren op eigenschap en sorteeropties opgeven
Het eigenlijke sorteren van de lijst met nummers wordt uitgevoerd door de klasse PlayList in de methode `sortList()`:

```
/**
 * 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;
}
```

Bij het sorteren op titel of artiest is alfabetisch sorteren handig, maar bij het sorteren op jaar is numeriek sorteren logischer. Met de instructie `switch` wordt de juiste sorteeroptie gedefinieerd, opgeslagen in de variabele `sortOptions`, op basis van de waarde die is vastgelegd in de parameter `sortProperty`. Opnieuw worden hier de benoemde leden van de opsomming gebruikt om onderscheid te maken tussen de eigenschappen (in plaats van hard gecodeerde waarden).

Wanneer de sorteereigenschap en sorteeropties zijn bepaald, wordt de array `_songs` pas echt gesorteerd door de methode `sortOn()` aan te roepen, waarbij die twee waarden als parameter worden doorgegeven. De huidige sorteereigenschap wordt vastgelegd, evenals het feit dat de lijst met nummers nu is gesorteerd.

Arrayelementen omzetten in een tekenreeks met een teken als scheidingsteken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naast het gebruik van een array om de lijst met nummers bij te houden in de klasse PlayList, worden in dit voorbeeld ook arrays gebruikt in de klasse Song als hulpmiddel bij het beheer van de lijst met genres waartoe een bepaald nummer behoort. Bekijk het volgende fragment uit de definitie van de klasse 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(";");
}
```

Wanneer een nieuwe instantie Song wordt gemaakt, wordt de parameter `genres` voor het opgeven van het genre (of de genres) van het nummer gedefinieerd als een instantie van het type Array. Dit maakt het gemakkelijker om meerdere genres te groeperen in één variabele die aan de constructor kan worden doorgegeven. Intern houdt de klasse Song de genres echter bij in de variabele van het type private `_genres` als instantie van het type String met een puntkomma als scheidingsteken. De parameter Array wordt omgezet in een tekenreeks met een puntkomma als scheidingsteken door de bijbehorende methode `join()` aan te roepen met de letterlijke tekenreekswaarde `";"` als opgegeven scheidingsteken.

Op dezelfde manier kunnen met de accessors van het type `genres` genres worden ingesteld of opgevraagd als Array:

```
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(";");
}
```

De accessor `genresset` gedraagt zich op precies dezelfde manier als de constructor: hij accepteert een Array en roept de methode `join()` aan voor de omzetting in een tekenreeks met een puntkomma als scheidingsteken. De accessor `get` voert een tegenovergestelde bewerking uit: de methode `split()` van de variabele `_genres` wordt aangeroepen, waardoor de String wordt opgedeeld in een array van waarden met het opgegeven scheidingsteken (dezelfde letterlijke tekenreekswaarde `";"` als daarvoor).

Hoofdstuk 4: Fouten afhandelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Foutafhandeling houdt in dat u logica in uw toepassing opneemt zodat de toepassing kan reageren op een fout of deze kan oplossen. Fouten worden gegenereerd wanneer een toepassing wordt gecompileerd of wanneer een gecompileerde toepassing wordt uitgevoerd. Wanneer uw toepassing een fout afhandelt, gebeurt er dus *iets* in reactie op een fout die optreedt, dit in tegenstelling tot geen enkele reactie en een ongemerkt mislukken van het proces dat de fout veroorzaakt. Bij een correct gebruik kan foutafhandeling uw toepassing en de gebruikers ervan beschermen tegen onverwacht gedrag.

Foutafhandeling is echter een breed begrip, waaronder vele soorten fouten vallen die kunnen optreden tijdens het compileren of uitvoeren van een toepassing. Deze discussie gaat over de afhandeling van uitvoerfouten, de verschillende typen fouten die gegenereerd kunnen worden en de voordelen van het foutafhandelingssysteem van ActionScript 3.0.

Basisbeginselen van foutafhandeling

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een uitvoeringsfout is iets dat verkeerd gaat in uw ActionScript-code waardoor de ActionScript-inhoud niet verder kan worden uitgevoerd. Als u wilt dat uw ActionScript-code probleemloos werkt voor gebruikers, moet u code aan uw toepassing toevoegen die ervoor zorgt dat de fout wordt afgehandeld. U kunt code schrijven die ervoor zorgt dat de fout wordt opgelost, dat een correcte omweg wordt gezocht of dat de gebruiker minstens een foutmelding krijgt. Dit hele proces wordt *foutafhandeling* genoemd.

Foutafhandeling is echter een breed begrip, waaronder vele soorten fouten vallen die kunnen optreden tijdens het compileren of uitvoeren van een toepassing. Fouten tijdens het compileren zijn meestal gemakkelijker te vinden. U moet ze oplossen om het maken van een SWF-bestand naar behoren af te ronden.

Uitvoeringsfouten zijn soms moeilijker op te sporen, omdat die zich pas voordoen wanneer de foute code echt wordt uitgevoerd. Als een segment in uw programma verschillende codevertakkingen bevat, zoals een `if..then..else`-instructie, moet u elke mogelijke voorwaarde testen met alle mogelijke invoerwaarden die echte gebruikers zouden kunnen gebruiken, om te bevestigen dat uw code foutloos is.

Uitvoeringsfouten kunnen in twee categorieën worden verdeeld: *programmafouten* zijn vergissingen in uw ActionScript-code (u gebruikt bijvoorbeeld het verkeerde gegevenstype voor een parameter van een methode) en *logische fouten* zijn vergissingen in de logica (het controleren van de gegevens en verwerken van waarden) van uw programma (u gebruikt bijvoorbeeld een verkeerde formule voor het berekenen van de rente in een banktoepassing). Beide soorten fouten kunnen vaak tijdig worden opgespoord en verholpen door de toepassing goed te testen.

Idealiter verwijdert u alle fouten voordat uw toepassing de eindgebruikers bereikt. Niet alle fouten kunnen echter worden voorzien of voorkomen. Stel bijvoorbeeld dat uw ActionScript-toepassing informatie uit een bepaalde website opvraagt, zonder dat u zelf enige controle over die website hebt. Als de website dan even niet beschikbaar is, werkt dat deel van uw toepassing dat afhankelijk is van die externe gegevens, niet goed. Het belangrijkste aspect van foutafhandeling is dat uw toepassing voorbereid is op deze onbekende gevallen en ze netjes kan afhandelen. Gebruikers moeten gewoon door kunnen gaan met het gebruik van uw toepassing of op zijn minst een duidelijk foutbericht krijgen waarom de toepassing niet meer werkt.

Uitvoeringsfouten worden in ActionScript op twee manieren vertegenwoordigd:

- Foutklassen: aan veel fouten is een foutklasse gekoppeld. Wanneer een fout optreedt, maakt de Flash runtime (Flash Player of Adobe AIR) een instantie van de specifieke foutklasse die bij die specifieke fout hoort. Aan de hand van de informatie in dat foutobject kan uw code zorgen voor de juiste reactie op de fout.
- Foutgebeurtenissen: soms treedt een fout op wanneer de Flash runtime normaal gesproken een gebeurtenis zou activeren. In dergelijke gevallen wordt er een foutgebeurtenis geactiveerd. Aan elke foutgebeurtenis is een klasse gekoppeld. De Flash-runtime geeft een instantie van die klasse door aan de methoden die op de foutgebeurtenis zijn geabonneerd.

Zie de vermelding van de methode in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform als u wilt weten of een bepaalde methode een fout of een foutgebeurtenis kan activeren.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het programmeren van foutafhandelingsroutines:

Asynchroon Een programmaopdracht, zoals een methodeaanroep, zonder onmiddellijk resultaat. In plaats daarvan heeft het resultaat (of de fout) de vorm van een gebeurtenis.

Afvangen Wanneer een uitzondering (een uitvoeringsfout) optreedt en uw code wordt zich hiervan bewust, wordt gezegd dat de code die uitzondering *afvangt*. Nadat de uitzondering is afgevangen, brengt Flash de andere ActionScript-code niet meer op de hoogte van de uitzondering.

Foutopsporingsversie Een speciale versie van Flash, zoals de foutopsporingsversie van Flash Player of AIR Debug Launcher (ADL), die code bevat waarmee gebruikers van runtimefouten op de hoogte worden gesteld. In de standaardversie van Flash Player of Adobe AIR (die de meeste gebruikers hebben) worden fouten die niet door uw ActionScript-code worden afgehandeld, gewoon genegeerd. In de foutopsporingsversies (die worden meegeleverd met Adobe Flash CS4 Professional en Adobe Flash Builder) wordt een waarschuwing weergegeven wanneer een niet-afgehandelde fout optreedt.

Uitzondering Een fout die optreedt wanneer een toepassing wordt uitgevoerd en die Flash niet zelfstandig kan oplossen.

Opnieuw genereren Wanneer uw code een uitzondering afvangt, worden andere objecten niet meer door Flash op de hoogte gesteld van de uitzondering. Als het belangrijk is dat andere objecten de uitzondering ontvangen, moet uw code de uitzondering *opnieuw genereren* om het meldingsproces opnieuw te beginnen.

Synchroon Een programmaopdracht, zoals een methodeaanroep, met een onmiddellijk resultaat (of met een onmiddellijk gegenereerde fout). Hierdoor kan de reactie worden gebruikt binnen hetzelfde codeblok.

Genereren Het op de hoogte brengen van Flash (en daardoor ook andere objecten en ActionScript-code) van het feit dat er zich een fout heeft voorgedaan, wordt een fout *genereren* genoemd.

Typen fouten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij het ontwikkelen en uitvoeren van toepassingen komt u verschillende soorten fouten en uiteenlopende terminologie tegen. De belangrijkste fouttypen en fouttermen worden hierna besproken:

- *Compilatiefouten* worden door de ActionScript-compiler gegenereerd tijdens het compileren van de code. Compilatiefouten doen zich ook wanneer syntaxisproblemen in uw code het onmogelijk maken om de definitieve toepassing te maken.
- *Uitvoeringsfouten* treden op wanneer u uw toepassing uitvoert na het compileren. Uitvoerfouten zijn fouten die optreden terwijl een SWF-bestand in een Flash runtime (zoals Adobe Flash Player of Adobe AIR) wordt afgespeeld. Meestal handelt u uitvoeringsfouten direct af door deze aan de gebruiker te melden en stappen te nemen om de toepassing aan de gang te houden. Als de fout fataal is (er kan bijvoorbeeld geen verbinding met een externe website worden gemaakt of de benodigde gegevens kunnen niet worden geladen), kunt u foutafhandeling gebruiken om de toepassing netjes te laten afsluiten.
- *Synchrone fouten* zijn uitvoerfouten die optreden op het moment dat een functie wordt aangeroepen, bijvoorbeeld wanneer u een specifieke methode probeert te gebruiken terwijl het aan de methode doorgegeven argument ongeldig is. Er wordt dan een uitzondering gegenereerd door de Flash-runtime. De meeste fouten zijn synchroon, dus bij het uitvoeren van een instructie, en het verloop wordt meteen doorgegeven aan de meest toepasselijke instructie van het type `catch`.

  Het volgende codefragment bijvoorbeeld genereert een uitvoeringsfout omdat de methode `browse()` niet wordt aangeroepen voordat het programma probeert een bestand te uploaden:

  ```
  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.
  }
  ```

  In dit geval wordt een synchrone uitvoeringsfout gegenereerd omdat Flash Player heeft vastgesteld dat de methode `browse()` niet is aangeroepen voordat werd geprobeerd het bestand te uploaden.

  Zie “Synchrone fouten in een toepassing afhandelen” op pagina 61 voor meer informatie over synchrone foutafhandeling.

- *Asynchronefouten* zijn runtimefouten die optreden buiten de normale programmastroom. Deze genereren gebeurtenissen en gebeurtenislisteners vangen deze af. Een asynchrone bewerking houdt in dat een functie een bewerking begint, maar niet wacht tot de bewerking klaar is. U kunt een listener voor een foutgebeurtenis maken die wacht tot de toepassing of gebruiker begint met een bewerking. Als deze bewerking mislukt, vangt u de fout af met een gebeurtenislistener en reageert u op de foutgebeurtenis. Vervolgens roept de gebeurtenislistener een gebeurtenishandlerfunctie aan om op een zinvolle manier op de foutgebeurtenis te reageren. De gebeurtenishandler kan bijvoorbeeld een dialoogvenster openen waarin de gebruiker wordt gevraagd de fout op te lossen.

Bekijk het eerder gegeven voorbeeld met de synchrone fout bij het uploaden van een bestand. Als u de methode `browse()` goed kunt aanroepen voordat u begint met het uploaden van een bestand, zou Flash Player verschillende gebeurtenissen verzenden. Bij het starten van de upload bijvoorbeeld wordt de gebeurtenis `open` verzonden. Wanneer het uploaden van het bestand goed is afgerond, wordt de gebeurtenis `complete` verzonden. Omdat gebeurtenisafhandeling asynchroon is (dat wil zeggen dat de afhandeling niet plaatsvindt op specifieke, bekende en vooraf bepaalde tijdstippen), moet u de methode `addEventListener()` gebruiken. Hierdoor wordt er geluisterd of deze specifieke gebeurtenissen plaatsvinden. Dit wordt in de volgende code getoond:

```
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...");
}
```

Zie “Reageren op foutgebeurtenissen en een specifieke status” op pagina 67 voor meer informatie over asynchrone foutafhandeling.

- *Niet-afgevangen uitzonderingen* zijn fouten die worden gegenereerd zonder bijbehorende logica (zoals een instructie `catch`) om erop te reageren. Als uw toepassing een fout genereert en er is op het huidige of hogere niveau geen passende instructie van het type `catch` of gebeurtenishandler aanwezig om de fout af te handelen, wordt de fout beschouwd als een niet-afgevangen uitzondering.

  Wanneer er een niet-afgevangen fout optreedt, wordt door de runtime de gebeurtenis `uncaughtError` verzonden. Deze gebeurtenis wordt ook wel een "globale fouthandler" genoemd. Deze gebeurtenis wordt verzonden door het object UncaughtErrorEvents van SWF, dat beschikbaar is via de eigenschap `LoaderInfo.uncaughtErrorEvents`. Als er geen listeners zijn geregistreerd voor de gebeurtenis `uncaughtError`, worden niet-afgevangen fouten tijdens runtime genegeerd. Zolang de fout SWF niet onderbreekt, wordt er doorgegaan met de uitvoering.

  Bij niet-afgevangen fouten verzenden foutopsporingsversies van de Flash-runtime niet alleen de gebeurtenis `uncaughtError`, maar reageren ze ook door het huidige script te beëindigen.· Vervolgens wordt de niet-afgevangen fout weergegeven in de uitvoer van een `trace`-instructie of wordt het foutbericht naar een logbestand geschreven. Als het uitzonderingsobject een instantie van de Error-klasse of een van de subklassen ervan is, worden stacktracegegevens ook weergegeven in de uitvoer. Zie “Werken met de foutopsporingsversies van Flash runtime” op pagina 60 voor meer informatie over het gebruik van de foutopsporingsversie van Flash Player.

  Opmerking: *Als tijdens de verwerking van een uncaughtError-gebeurtenis een foutgebeurtenis wordt gegenereerd door een uncaughtError-handler, wordt de gebeurtenishandler meerdere malen aangeroepen. Dit leidt tot een oneindige lus van uitzonderingen. U kunt een dergelijk scenario beter vermijden.*

Foutafhandeling in ActionScript 3.0

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat veel toepassingen kunnen draaien zonder logica voor de afhandeling van fouten, zijn ontwikkelaars geneigd om het inbouwen van foutafhandeling op de lange baan te schuiven. Maar een toepassing zonder foutafhandeling kan snel crashen of voor ergernis bij de gebruiker zorgen omdat iets niet werkt zoals verwacht. ActionScript 2.0 heeft een klasse van het type Error die het mogelijk maakt om logica toe te voegen aan aangepaste functies voor het genereren van uitzonderingenmet een bepaalde foutmelding. Omdat een correcte foutafhandeling cruciaal is voor een gebruiksvriendelijke toepassing, bevat ActionScript 3.0 een uitgebreide architectuur voor het afvangen van fouten.

Opmerking: *Hoewel de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform beschrijvingen bevat over de uitzonderingen die door vele methoden worden gegenereerd, is het mogelijk dat niet alle uitzonderingen voor elke methode hierin zijn opgenomen. Ook al bevat de beschrijving informatie over enkele uitzonderingen die door de methode kunnen worden gegenereerd, kan een methode ook uitzonderingen genereren voor syntaxisfouten of andere problemen die niet uitdrukkelijk in de beschrijving van de methode staan vermeld.*

Elementen van de foutafhandeling van ActionScript 3.0

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 bevat veel gereedschappen voor de afhandeling van fouten, waaronder:

- Foutklassen. ActionScript 3.0 bevat een groot aantal Error-klassen om het bereik uit te breiden van situaties waardoor foutobjecten kunnen worden gegenereerd. Elke foutklasse helpt toepassingen bij het afhandelen van en reageren op specifieke foutomstandigheden in verband met systeemfouten (zoals bij een MemoryError), coderingsfouten (zoals bij een ArgumentError), netwerk- en communicatiefouten (zoals bij een URIError) en andere omstandigheden. Zie "De foutklassen vergelijken" op pagina 70 voor meer informatie over elke klasse.
- Minder onopgemerkte fouten. In eerdere versies van Flash Player werden fouten alleen gegenereerd en gemeld als u uitdrukkelijk de instructie `throw` gebruikte. Bij Flash Player 9 en hoger worden uitvoerfouten gegenereerd door native ActionScript-methoden en -eigenschappen. Door deze fouten kunt u uitzonderingen effectiever afhandelen op het moment dat deze zich voordoen. Vervolgens kunt u op elke uitzondering afzonderlijk reageren.
- Duidelijke foutmeldingen tijdens het opsporen van fouten. Bij het gebruik van de foutopsporingsversie van Flash-runtime worden bij problematische code of omstandigheden duidelijke foutmeldingen gegenereerd. Aan de hand hiervan kunt u gemakkelijker terugvinden waarom een bepaald codeblok niet goed werkt. Door deze berichten verloopt het opsporen van fouten efficiënter. Zie "Werken met de foutopsporingsversies van Flash runtime" op pagina 60 voor meer informatie.
- Nauwkeurige fouten maken nauwkeurige foutmeldingen voor de gebruiker mogelijk tijdens de uitvoering. Bij eerdere versies van Flash Player retourneerde de methode `FileReference.upload()` een Booleaanse waarde `false` als het aanroepen van `upload()` mislukte. Dit wees dan op vijf mogelijke fouten. Als een fout optreedt bij het aanroepen van de methode `upload()` in ActionScript 3.0, kunt u aan de hand van vier specifieke fouten nauwkeurigere foutmeldingen voor de eindgebruikers laten weergeven.
- Verfijnde foutafhandeling. Voor tal van veelvoorkomende situaties worden specifieke fouten gegenereerd. In ActionScript 2.0 heeft de eigenschap `name` bijvoorbeeld de waarde `null` voordat een FileReference-object is gevuld (dus voordat u de eigenschap `name` kunt gebruiken of weergeven, moet u ervoor zorgen dat de waarde is ingesteld en niet `null` is). Als u in ActionScript 3.0 probeert de eigenschap `name` te benaderen voordat die is gevuld, genereert Flash Player of AIR een IllegalOperationError, waarbij wordt gemeld dat de waarde niet is ingesteld. Vervolgens kunt u de fout afhandelen met blokken van het type `try..catch..finally`. Zie "Instructies van het type try..catch..finally gebruiken" op pagina 61 voor meer informatie.

- Geen significante nadelen voor de prestaties. Het gebruik van blokken van het type `try..catch..finally` voor de afhandeling van fouten vergt weinig of geen extra bronnen ten opzichte van eerdere versies van ActionScript.
- De klasse ErrorEvent maakt het mogelijk om listeners te maken voor specifieke asynchrone foutgebeurtenissen. Zie "Reageren op foutgebeurtenissen en een specifieke status" op pagina 67 voor meer informatie.

Strategieën voor de foutafhandeling

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Ook als u geen logica voor de afhandeling van fouten aan uw code toevoegt, werkt uw toepassing normaal zolang er zich geen problemen voordoen. Maar als fouten niet actief worden afgehandeld en uw toepassing komt wel een probleem tegen, komen uw gebruikers nooit te weten waarom de toepassing het op dat specifieke moment niet meer doet.

U kunt de foutafhandeling in uw toepassing op verschillende manieren aanpakken. Hieronder staan de drie belangrijkste opties voor de foutafhandeling:

- Gebruik instructies van het type `try..catch..finally`. Hiermee worden synchrone fouten afgevangen op het moment waarop ze zich voordoen. U kunt uw instructies nesten in een hiërarchie om uitzonderingen af te vangen op verschillende niveaus van de uitgevoerde code. Zie "Instructies van het type try..catch..finally gebruiken" op pagina 61 voor meer informatie.
- Maak uw eigen aangepaste foutobjecten. Met de klasse Error kunt u uw eigen aangepaste foutobjecten maken om specifieke bewerkingen in uw toepassing bij te houden waarvoor geen ingebouwde fouttypen beschikbaar zijn. Vervolgens kunt u de instructies `try..catch..finally` in uw aangepaste foutobjecten gebruiken. Zie "Aangepaste foutklassen maken" op pagina 66 voor meer informatie.
- Schrijf gebeurtenislisteners en -handlers om te reageren op foutgebeurtenissen. Aan de hand van deze strategie kunt u algemene fouthandlers maken waarmee u soortgelijke gebeurtenissen kunt afhandelen zonder dat u veel code hoeft te dupliceren in de blokken `try..catch..finally`. Op deze manier zult u ook sneller asynchrone fouten kunnen afvangen. Zie "Reageren op foutgebeurtenissen en een specifieke status" op pagina 67 voor meer informatie.

Werken met de foutopsporingsversies van Flash runtime

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Adobe heeft speciale uitgaven van Flash runtime voor ontwikkelaars als hulp bij de foutopsporing. Wanneer u Adobe Flash Professional of Adobe Flash Builder installeert, ontvangt u daarbij ook de foutopsporingsversie van Flash Player. U ontvangt ook de foutopsporingsversie van Adobe AIR, ADL genoemd, wanneer u een van deze hulpprogramma's installeert of als onderdeel van de SDK voor Adobe AIR.

Er is een belangrijk verschil in de manier waarop de foutopsporingsversies en de releaseversies van Flash Player en Adobe AIR fouten aangeven. De foutopsporingsversies laten het fouttype zien (zoals een algemene fout, een IOError of een EOFError), het foutnummer en een leesbare foutmelding. De releaseversies laten alleen het fouttype en -nummer zien. Neem bijvoorbeeld de volgende code:

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

Als de methode `readBoolean()` een EOFError genereert in de foutopsporingsversie van Flash Player, wordt de volgende melding weergegeven in het tekstveld `tf`: “EOFError: Error #2030: End of file was encountered.”

Dezelfde code in een releaseversie van Flash Player of Adobe AIR zou de volgende tekst weergeven: “EOFError: Error #2030.”

Opmerking: *De foutopsporingsspelers zenden de gebeurtenis "allComplete" uit. Zorg er daarom voor dat u de naam “allComplete” niet gebruikt voor aangepaste gebeurtenissen. Als u dit toch doet, treedt onvoorspelbaar gedrag op tijdens de foutopsporing.*

Om de resources en omvang van Flash Player tot een minimum beperkt te houden, is in de releaseversies geen tekst voor de foutmeldingen aanwezig. U kunt het nummer van de fout opzoeken in de documentatie (in de bijlagen van de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform) om de bijbehorende foutmelding te lezen. U kunt de fout ook reproduceren met de foutopsporingsversies van Flash Player en AIR om de volledige melding te zien.

Synchrone fouten in een toepassing afhandelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Logica voor de afhandeling van synchrone fouten komt het meest voor. Dit houdt in dat u in uw code instructies gebruikt om synchrone fouten tijdens de uitvoering af te vangen. Door dit soort foutafhandeling weet de toepassing dat er iets mis is en kunnen uitvoeringsfouten worden hersteld. De logica voor het afvangen van een synchrone fout bevat instructies van het type `try..catch..instructies van het type finally`, die letterlijk een bewerking proberen, eventuele foutreacties van de Flash runtime afvangen en ten slotte een andere bewerking uitvoeren om de mislukte bewerking af te handelen.

Instructies van het type try..catch..finally gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u met synchrone uitvoeringsfouten werkt, gebruikt u de instructies `try..catch..finally` om fouten af te vangen. Wanneer een uitvoeringsfout optreedt, genereert Flash runtime een uitzondering. Dit betekent dat Flash runtime de normale uitvoering opschort en een speciaal object van het type Error maakt. Dit object wordt vervolgens gegenereerd voor het eerste beschikbare blok `catch`.

De instructie `try` omsluit instructies die aanleiding kunnen geven tot fouten. U moet de instructie `catch` altijd gebruiken met een instructie van het type `try`. Als een fout wordt gevonden in een van de instructies in het blok van de instructie `try`, worden de `catch`-instructies die bij die `try`-instructie horen uitgevoerd.

De instructie `finally` omsluit instructies die hoe dan ook worden uitgevoerd, of er nu wel of niet een fout optreedt in het blok `try`. Als er geen fout optreedt, worden de instructies in het blok `finally` uitgevoerd na afloop van de instructies uit het blok `try`. Als er wel een fout optreedt, wordt eerst de desbetreffende `catch`-instructie uitgevoerd en daarna komen de instructies uit het blok `finally`.

In de volgende code ziet u de syntaxis voor het gebruik van de instructies `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.
}
```

Elke instructie van het type `catch` identificeert een specifiek type uitzondering die de instructie afhandelt. Met `catch` kunnen alleen foutklassen worden opgegeven die een subklasse zijn van de klasse Error. Elke `catch`-instructie wordt in volgorde gecontroleerd. Alleen de eerste `catch`-instructie die met het type gegenereerde fout overeenkomt, wordt uitgevoerd. Met andere woorden: als u eerst de klasse Error van het hogere niveau controleert en vervolgens een subklasse van de klasse Error, komt alleen de klasse Error van het hogere niveau overeen. Dit wordt in de volgende code geïllustreerd:

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

De vorige code geeft de volgende uitvoer weer:

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

Om de ArgumentError goed te kunnen afvangen moet u ervoor zorgen dat de meest specifieke fouttypen als eerste worden genoemd. De meer algemene fouttypen vermeldt u dan later, zoals in de volgende code wordt getoond:

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

Diverse methoden en eigenschappen in de ActionScript-API genereren uitvoeringsfouten als ze fouten tegenkomen tijdens de uitvoering. Zo genereert de methode `close()` in de klasse Sound een IOError als de methode niet in staat is om de audiostream te sluiten. Dit wordt getoond in de volgende code:

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

Naarmate u vertrouwder raakt met de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform, zult u merken welke methoden uitzonderingen genereren. Dit wordt in de beschrijving van de methoden uitgelegd.

De instructie throw

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash runtime genereert uitzonderingen wanneer deze fouten tijdens de uitvoering fouten in uw toepassing tegenkomen. Bovendien kunt u uitdrukkelijk zelf uitzonderingen genereren met behulp van de instructie `throw`. Wanneer u uitdrukkelijk zelf fouten genereert, raadt Adobe u aan om dit te doen met instanties van de klasse Error of een van de subklassen daarvan. De volgende code bevat een voorbeeld van een instructie van het type `throw` waarmee een instantie van de klasse Error wordt gegenereerd (`MyErr`) en waarmee uiteindelijk een functie wordt aangeroepen (`myFunction()`) als reactie na het genereren van de fout:

```
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.
}
```

De instructies `catch` zijn zo geordend dat de meest specifieke gegevenstypen als eerste komen. Als de instructie `catch` voor het gegevenstype Number als eerste voorkomt, wordt de instructie catch voor het gegevenstype uint noch de instructie catch voor het gegevenstype int ooit uitgevoerd.

Opmerking: *In de programmeertaal Java moeten alle functies die een uitzondering kunnen genereren dit feit eerst declareren. Daarbij moeten de mogelijk gegenereerde uitzonderingsklassen worden vermeld in een clausule van het type* `throws` *die aan de functiedeclaratie is gekoppeld. Bij ActionScript is het niet verplicht om de uitzonderingen te declareren die door een functie worden gegenereerd.*

Een eenvoudige foutmelding weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een van de grootste voordelen van het nieuwe model voor uitzonderingen en foutgebeurtenissen is dat het nu mogelijk is om de gebruikers te laten weten dat een actie is mislukt en waarom dat is gebeurd. Het is uw taak om code te schrijven voor het weergeven van de foutmelding en voor het aanbieden van opties in reactie daarop.

De volgende code bevat een enkelvoudige instructie `try..catch` om de fout in een tekstveld weer te geven:

```
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);
            }
        }
    }
}
```

Met een grotere verscheidenheid aan foutklassen en ingebouwde compilatiefouten verschaft ActionScript 3.0 meer informatie dan vorige versies van ActionScript over het waarom van een probleem. Met deze informatie kunt u stabielere toepassingen maken met een betere foutafhandeling.

Fouten opnieuw genereren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij het maken van toepassingen kan het gebeuren dat u een fout opnieuw moet genereren als het niet lukt om de fout goed af te handelen. De volgende code bevat bijvoorbeeld een genest blok `try..catch`, waarmee een aangepaste ApplicationError opnieuw wordt gegenereerd als het geneste blok `catch` de fout niet kan afhandelen:

```
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);
}
```

De uitvoer van het vorige fragment wordt dan dit:

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

Het geneste blok `try` genereert een aangepaste ApplicationError die door het volgende blok `catch` wordt afgevangen. Dit geneste blok `catch` kan proberen de fout af te handelen en als dit niet lukt het ApplicationError-object genereren voor het omsluitende blok `try..catch`.

Aangepaste foutklassen maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript kunt u een van de standaardfoutklassen uitbreiden met uw eigen gespecialiseerde foutklassen. Er zijn verschillende redenen om uw eigen foutklassen te maken:

- Om specifieke fouten of groepen fouten te identificeren die uniek zijn voor uw toepassing.

  U kunt bijvoorbeeld verschillende acties maken voor fouten die door uw eigen code worden gegenereerd, naast die van Flash-runtime. U kunt een subklasse van de klasse Error maken om het type foutgegevens vast te leggen in blokken van het type `try..catch`.

- Om unieke weergavemogelijkheden te creëren voor fouten die uw toepassing genereert.

  U kunt bijvoorbeeld een nieuwe methode `toString()` maken die uw foutmelding op een bepaalde manier opmaakt. U kunt ook een methode `lookupErrorString()` definiëren die uitgaat van de foutcode en vervolgens de juiste melding opvraagt op basis van de taalvoorkeur van de gebruiker.

Een gespecialiseerde foutklasse moet een uitbreiding zijn op de kernklasse Error van ActionScript. Hierna volgt een voorbeeld van een gespecialiseerde klasse AppError om de klasse Error uit te breiden:

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

In het volgende voorbeeld wordt getoond hoe u AppError kunt gebruiken in uw project:

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

Opmerking: *Als u de methode `Error.toString()` in uw subklassen wilt overschrijven, moet u er één parameter `...(rest)` aan geven. De ECMAScript-taalspecificatie waarop ActionScript 3.0 is gebaseerd, definieert de methode `Error.toString()` op deze wijze, en ActionScript 3.0 definieert deze op dezelfde wijze voor achterwaartse compatibiliteit. Dus wanneer u de methode `Error.toString()` overschrijft, moet u de parameters precies laten overeenkomen. Het heeft geen zin om tijdens runtime parameters aan uw methode `toString()` door te geven, omdat deze parameters dan worden genegeerd.*

Reageren op foutgebeurtenissen en een specifieke status

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een goed merkbare verbetering in de foutafhandeling van ActionScript 3.0 is de ondersteuning voor de afhandeling van foutgebeurtenissen om te kunnen reageren op asynchrone uitvoeringsfouten. (Zie "Typen fouten" op pagina 57 voor een definitie van asynchrone fouten.)

U kunt gebeurtenislisteners en -handlers maken om te reageren op foutgebeurtenissen. Veel klassen verzenden een foutgebeurtenis op dezelfde manier als andere gebeurtenissen. Een instantie van de klasse XMLSocket bijvoorbeeld verzendt normaal gesproken drie typen gebeurtenissen: `Event.CLOSE`, `Event.CONNECT` en `DataEvent.DATA`. Maar wanneer een probleem optreedt, kan de klasse XMLSocket de `IOErrorEvent.IOError` of de `SecurityErrorEvent.SECURITY_ERROR` verzenden. Zie "Gebeurtenissen afhandelen" op pagina 133 voor meer informatie over gebeurtenislisteners en -handlers.

Foutgebeurtenissen behoren tot een van de volgende twee categorieën:

- Foutgebeurtenissen als uitbreiding op de klasse ErrorEvent

  De klasse flash.events.ErrorEvent bevat de eigenschappen en methoden voor het beheer van Flash Player-uitvoeringsfouten met betrekking tot netwerk- en communicatiefouten. De klassen AsyncErrorEvent, IOErrorEvent en SecurityErrorEvent zijn een uitbreiding op de klasse ErrorEvent. Als u de foutopsporingsversie van de Flash-runtime gebruikt, wordt u tijdens runtime middels een dialoogvenster op de hoogte gebracht van foutgebeurtenissen zonder listenerfuncties die de speler tegenkomt.

- Foutgebeurtenissen op basis van een status

Foutgebeurtenissen op basis van een status houden verband met de eigenschappen `netStatus` en `status` van de netwerk- en communicatieklassen. Als Flash Player of Adobe AIR een probleem tegenkomt bij het lezen of schrijven van gegevens, wordt de waarde van de eigenschap `netStatus.info.level` of `status.level` (afhankelijk van het klassenobject dat u gebruikt) gewijzigd in `"error"`. U reageert op deze fout door na te gaan of de eigenschap `level` de waarde `"error"` bevat in uw gebeurtenishandlerfunctie.

Werken met foutgebeurtenissen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse ErrorEvent en de bijbehorende subklassen bevatten fouttypen voor de afhandeling van fouten die door Flash runtimes worden verzonden bij het lezen of schrijven van gegevens.

In het volgende voorbeeld wordt zowel een instructie `try..catch` als foutgebeurtenishandlers gebruikt om fouten weer te geven die tijdens het lezen naar een lokaal bestand zijn gesignaleerd. Met de opmerking 'hier uw foutafhandelingscode' wordt bedoeld dat u op die plaats uw eigen meer geavanceerde code voor de foutafhandeling kunt toevoegen om de gebruiker een aantal opties te bieden of om de fout anderszins automatisch af te handelen:

```
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
        }
    }
}
```

Werken met gebeurtenissen die een statuswijziging inhouden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash runtimes kunnen de waarde van de eigenschap `netStatus.info.level` of `status.level` dynamisch wijzigen voor klassen die de eigenschap `level` ondersteunen. De klassen met de eigenschappen `netStatus.info.level` zijn NetConnection, NetStream en SharedObject. De klassen met de eigenschap `status.level` zijn HTTPStatusEvent, Camera, Microphone en LocalConnection. U kunt een handlerfunctie schrijven om te reageren op een wijziging in de waarde `level` en om communicatiefouten bij te houden

In het volgende voorbeeld wordt de functie `netStatusHandler()` gebruikt om de waarde van de eigenschap `level` te testen. Als de eigenschap `level` aangeeft dat een fout is opgetreden, geeft de code de melding dat de videostream is mislukt.

```
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);
        }
    }
}
```

De foutklassen vergelijken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript bevat een aantal vooraf gedefinieerde Error-klassen. Maar u kunt dezelfde klassen ook gebruiken in uw eigen code. Er zijn twee hoofdtypen Error-klassen in ActionScript 3.0: ActionScript-kernklassen van het type Error en flash.error-pakketklassen van het type Error. Het pakket flash.error bevat extra klassen die dienen als hulpmiddel bij het ontwikkelen van ActionScript 3.0-toepassingen en bij het opsporen van fouten hierin.

Kernklassen van het type Error

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De kernklassen van het type Error bevatten onder meer de klassen Error, ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError en VerifyError. Elk van deze klassen bevindt zich in de naamruimte op hoofdniveau.

Klassennaam	Beschrijving	Opmerkingen
Fout	De klasse Error wordt gebruikt om uitzonderingen te genereren en vormt de basisklasse voor de overige uitzonderingsklassen die in ECMAScript zijn vastgelegd: EvalError, RangeError, ReferenceError, SyntaxError, TypeError en URIError.	De klasse Error dient als basisklasse voor alle runtimefouten en is de aanbevolen basisklasse voor al uw aangepaste foutklassen.
ArgumentError	De klasse ArgumentError vertegenwoordigt een fout die optreedt wanneer de parameterwaarden die in een functieaanroep worden aangeboden, niet overeenkomen met de parameters die voor die functie zijn gedefinieerd.	Voorbeelden van argumentfouten zijn: • De methode krijgt te weinig of te veel argumenten aangeleverd. • Het argument wordt geacht lid van een opsomming te zijn en is dat niet.
EvalError	Een uitzondering van het type EvalError wordt gegenereerd als parameters worden doorgegeven aan de constructor van de klasse Function of als de code van de gebruiker de functie `eval()` aanroept.	In ActionScript 3.0 is de ondersteuning voor de functie `eval()` verwijderd en wordt bij pogingen om de functie te gebruiken een fout gegenereerd. In eerdere versies van Flash Player werd de functie `eval()` gebruikt om variabelen, eigenschappen, objecten en filmclips te benaderen aan de hand van de naam.
RangeError	Een uitzondering van het type RangeError wordt gegenereerd als een numerieke waarde buiten het acceptabele bereik ligt.	Er wordt bijvoorbeeld een RangeError gegenereerd door de klasse Timer bij een negatieve of niet eindige vertraging. Er kan ook een RangeError worden gegenereerd als u een weergaveobject probeert toe te voegen op een ongeldige diepte.
ReferenceError	Een uitzondering ReferenceError wordt gegenereerd wanneer een verwijzing naar een ongedefinieerde eigenschap wordt opgegeven bij een verzegeld (niet-dynamisch) object. Versies van de ActionScript-compiler vóór ActionScript 3.0 genereerden geen fout wanneer een eigenschap werd benaderd die `undefined` was. ActionScript 3.0 genereert in deze situatie echter de uitzonderingsfout ReferenceError.	Uitzonderingen voor ongedefinieerde variabelen wijzen op mogelijke fouten, zodat u uw software kunt verbeteren. Als u echter niet gewend bent aan het initialiseren van uw variabelen, kan dit nieuwe ActionScript-gedrag enige aanpassing vergen aan de manier waarop u gewoonlijk code schrijft.
SecurityError	De uitzondering SecurityError wordt gegenereerd wanneer de beveiliging wordt geschonden en de toegang wordt geweigerd.	Voorbeelden van beveiligingsfouten zijn: • Ongeautoriseerde toegang tot een eigenschap of een aanroep van een methode die een beveiligingssandboxgrens heeft overschreden. • Er is geprobeerd toegang tot een URL te verkrijgen, dat niet is toegestaan door de beveiligingssandbox. • Er is geprobeerd een socketverbinding tot stand te brengen met een poort, maar het noodzakelijke socketbeleidsbestand was niet aanwezig. • Er is een poging gedaan toegang te krijgen tot de camera of de microfoon van de gebruiker, maar de toegang tot het apparaat is door de gebruiker geweigerd.
SyntaxError	Er wordt een uitzondering van het type SyntaxError gegenereerd wanneer in uw ActionScript-code een fout bij het parseren optreedt.	Een SyntaxError kan worden gegenereerd in de volgende omstandigheden: • ActionScript genereert SyntaxError-uitzonderingen wanneer een ongeldige reguliere expressie door de klasse RegExp wordt geparseerd. • ActionScript genereert SyntaxError-uitzonderingen wanneer ongeldige XML door de klasse XMLDocument wordt geparseerd.

Klassennaam	Beschrijving	Opmerkingen
TypeError	Er wordt een uitzondering TypeError gegenereerd wanneer het daadwerkelijke type van een operand verschilt van het verwachte type.	Een TypeError kan worden gegenereerd in de volgende omstandigheden: • Een feitelijke parameter van een functie of methode kon niet tot het formele parametertype worden gebracht. • Een waarde wordt toegewezen aan een variabele en kan niet tot het variabeletype worden gebracht. • De rechterkant van de operator `is` of `instanceof` is geen geldig type. • Het trefwoord `super` wordt ongeldig gebruikt. • Het opzoeken van een eigenschap resulteert in meer dan één binding en is dus dubbelzinnig. • Een methode wordt aangeroepen voor een incompatibel object. Er wordt bijvoorbeeld een TypeError-uitzondering gegenereerd als een methode van de klasse RegExp wordt samengevoegd met een algemeen object en vervolgens wordt aangeroepen.
URIError	De uitzondering URIError wordt gegenereerd wanneer een van de algemene URI-verwerkingsfuncties wordt gebruikt op een manier die niet compatibel is met de definitie ervan.	Een URIError kan worden gegenereerd in de volgende omstandigheden: Er is een ongeldige URI opgegeven voor een Flash Player API-functie die een geldige URI verwacht, zoals `Socket.connect()`.
VerifyError	Er wordt een uitzondering van het type VerifyError gegenereerd wanneer een onjuist geformuleerd of beschadigd SWF-bestand wordt aangetroffen.	Wanneer een SWF-bestand een ander SWF-bestand laadt, kan het bovenliggende SWF-bestand een VerifyError afvangen die door het geladen SWF-bestand is gegenereerd.

Klassen van het type Error uit het pakket flash.error

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het pakket flash.error bevat klassen van het type Error die worden beschouwd als onderdeel van de API van de Flash-runtime. In tegenstelling tot de beschreven klassen van het type Error communiceert het pakket flash.error foutgebeurtenissen die specifiek zijn voor Flash-runtimes (zoals Flash Player of Adobe AIR).

Klassennaam	Beschrijving	Opmerkingen
EOFError	Een uitzondering EOFError wordt gegenereerd wanneer u voorbij het einde van de beschikbare gegevens wilt lezen.	Er wordt bijvoorbeeld een EOFError gegenereerd wanneer een van de leesmethoden in de IDataInput-interface wordt aangeroepen en er onvoldoende gegevens beschikbaar zijn om aan de leesaanvraag te voldoen.
IllegalOperationError	De uitzondering IllegalOperationError wordt gegenereerd wanneer een methode niet is geïmplementeerd of de implementatie niet geldt voor het huidige gebruik.	Voorbeelden van uitzonderingen van het type IllegalOperationError: • Een basisklasse, zoals DisplayObjectContainer, biedt meer functionaliteit dan het werkgebied kan ondersteunen. Als u bijvoorbeeld een masker ophaalt of instelt in het werkgebied (met `stage.mask`), genereert de Flash-runtime een IllegalOperationError met de melding dat de klasse Stage deze eigenschap of methode niet implementeert. • Een subklasse overerft een methode die de subklasse niet nodig heeft en niet wil ondersteunen. • Er worden bepaalde toegankelijkheidsmethoden aangeroepen wanneer Flash Player wordt gecompileerd zonder toegankelijkheidsondersteuning. • Functies die alleen bestemd zijn voor ontwerpen, worden aangeroepen door een runtimeversie van Flash Player. • U probeert de naam in te stellen van een object dat op de tijdlijn is geplaatst.
IOError	De uitzondering IOError wordt gegenereerd wanneer een fout optreedt bij invoer of uitvoer.	U krijgt deze fout bijvoorbeeld wanneer een lees- of schrijfbewerking wordt uitgevoerd via een socket die niet is verbonden of waarvan de verbinding is verbroken.

Klassennaam	Beschrijving	Opmerkingen
MemoryError	De uitzondering MemoryError wordt gegenereerd wanneer een aanvraag voor geheugentoewijzing mislukt.	ActionScript Virtual Machine 2 stelt standaard geen limiet aan de hoeveelheid geheugen die door een ActionScript-programma kan worden toegewezen. Op een gewone desktopcomputer komen fouten met de geheugentoewijzing niet zo veel voor. Er wordt een fout gegenereerd wanneer het systeem niet in staat is om het geheugen toe te wijzen dat nodig is voor een bewerking. Daarom is deze uitzondering zeldzaam op een desktopcomputer, tenzij er extreem veel geheugen wordt gevraagd. Zo is een aanvraag voor 3 miljard bytes onmogelijk, omdat een 32-bits Microsoft® Windows®-programma niet meer dan 2 GB aan adresruimte kan benaderen.
ScriptTimeoutError	De uitzondering ScriptTimeoutError wordt gegenereerd wanneer een time-outinterval van 15 seconden voor het script wordt bereikt. Door de uitzondering ScriptTimeoutError af te vangen kunt u de time-out in het script netter afhandelen. Als er geen uitzonderingshandler beschikbaar is, wordt een dialoogvenster met een foutmelding weergegeven door de handler voor niet-afgevangen uitzonderingen.	Om te voorkomen dat een ontwikkelaar met minder fraaie bedoelingen de uitzondering afvangt en een oneindige lus creëert, kan alleen de eerste uitzondering van het type ScriptTimeoutError die tijdens de uitvoering van een bepaald script wordt gegenereerd, worden afgevangen. Wanneer deze uitzondering daarna nog een keer optreedt, is afvangen niet meer mogelijk. De uitzondering gaat onmiddellijk naar de handler voor niet-afgevangen uitzonderingen.
StackOverflowError	De uitzondering StackOverflowError wordt gegenereerd wanneer de stack die voor het script beschikbaar is, overloopt.	Een uitzondering van het type StackOverflowError kan erop wijzen dat een oneindige herhaling is opgetreden.

Voorbeeld van foutafhandeling: CustomErrors-toepassing

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De CustomErrors-toepassing demonstreert technieken voor het werken met aangepaste fouten bij het maken van een toepassing. Deze technieken zijn:

- Een XML-pakket valideren
- Een aangepaste fout schrijven
- Aangepaste fouten genereren
- De gebruikers laten weten dat een fout is gegenereerd

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de bestanden voor de CustomErrors-toepassing in de map Samples/CustomError. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
CustomErrors.mxml of CustomErrors.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML)
com/example/programmingas3/errors/ApplicationError.as	Een klasse die ook fungeert als basisfoutklasse voor de klassen FatalError en WarningError.
com/example/programmingas3/errors/FatalError.as	Een klasse die een door de toepassing gegenereerde fout van het type FatalError definieert. Deze klasse breidt de aangepaste klasse ApplicationError uit.
com/example/programmingas3/errors/Validator.as	Een klasse die één methode definieert voor de validatie van een door de gebruiker opgegeven employee-XML-pakket.
com/example/programmingas3/errors/WarningError.as	Een klasse die een door de toepassing gegenereerde fout van het type WarningError definieert. Deze klasse breidt de aangepaste klasse ApplicationError uit.

Overzicht CustomErrors-toepassing

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de toepassing wordt geladen, wordt de methode `initApp()` aangeroepen voor Flex-toepassingen of wordt de tijdlijncode (niet functioneel) uitgevoerd voor Flash Professional-toepassingen. Deze code definieert een voorbeeld-XML-pakket dat door de klasse Validator wordt gecontroleerd. De volgende code wordt uitgevoerd:

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

Het XML-pakket wordt later in het werkgebied weergegeven in een instantie van een TextArea-component. Hierdoor kunt u het XML-pakket wijzigen voordat u probeert het opnieuw te valideren.

Wanneer de gebruiker op de knop Valideren klikt, wordt de methode `validateData()` aangeroepen. Deze methode valideert het employee-XML-pakket met de methode `validateEmployeeXML()` uit de klasse Validator. De volgende code toont de methode `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);
    }
}
```

Eerst wordt een tijdelijk XML-object gemaakt met behulp van de inhoud van de instantie `xmlText` van de TextArea-component. Vervolgens wordt de methode `validateEmployeeXML()` uit de aangepaste klasse Validator aangeroepen (com.example.programmingas3/errors/Validator.as) en wordt het tijdelijke XML-object doorgegeven als parameter. Als het XML-pakket geldig is, geeft de instantie `status` van de Label-component een succesbericht weer en wordt de toepassing afgesloten. Als de methode `validateEmployeeXML()` een aangepaste fout genereert (dat wil zeggen, dat een FatalError, WarningError of een algemene Error optreedt), wordt de desbetreffende instructie `catch` uitgevoerd die de methode`showFatalError()`, `showWarningError()` of `showGenericError()` aanroept. Deze methoden geven elk het toepasselijke bericht weer in een tekstgebied, `statusText`, om de gebruiker op de hoogte te stellen van de specifieke fout. Elke methode werkt ook de instantie `status` van de Label-component bij met een specifiek bericht.

Als een fatale fout optreedt tijdens een poging om het employee-XML-pakket te valideren, wordt de foutmelding weergegeven in het tekstgebied `statusText` en worden de instantie `xmlText` van de TextArea-component en de instantie `validateBtn` van de Button-component uitgeschakeld, zoals in de volgende code wordt getoond:

```
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();
}
```

Als een waarschuwingsfout in plaats van een fatale fout optreedt, wordt de foutmelding weergegeven in de TextArea-instantie `statusText`, maar in dit geval worden de `xmlText`-instanties van de TextField- en Button-component niet uitgeschakeld. De methode `showWarningError()` geeft het aangepaste foutbericht weer in het tekstgebied `statusText`. In deze melding wordt de gebruiker ook gevraagd of hij of zij wil doorgaan met de XML-validatie of dat het script moet worden geannuleerd. Het volgende fragment toont de methode `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;
}
```

Wanneer de gebruiker op de knop Ja of Nee klikt, wordt de methode `closeHandler()` aangeroepen. Het volgende fragment toont de methode `closeHandler()`:

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

Als de gebruiker ervoor kiest het script te annuleren door op Ja te klikken, wordt een FatalError gegenereerd, waardoor de toepassing wordt afgesloten.

Een aangepaste validator maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De aangepaste klasse Validator bevat één methode: `validateEmployeeXML()`. De methode `validateEmployeeXML()` werkt met één argument (`employee`). Dit is het XML-pakket dat u wilt valideren. De methode `validateEmployeeXML()` ziet er als volgt uit:

```
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);
    }
}
```

Om te worden gevalideerd moet een employee tot één kostenplaats behoren (absoluut niet meer dan één kostenplaats). Als de employee tot geen enkele kostenplaats behoort, genereert de methode een FatalError, die omhoog loopt naar de methode `validateData()` in het hoofdtoepassingsbestand. Als de employee tot meer dan één kostenplaats behoort, wordt een WarningError gegenereerd. De definitieve controle in de XML-validator is dat de gebruiker precies één sofinummer heeft gedefinieerd (de node `ssn` in het XML-pakket). Als er niet precies één node van het type `ssn` is, wordt een FatalError gegenereerd.

U kunt extra controles aan de methode `validateEmployeeXML()` toevoegen, bijvoorbeeld om ervoor te zorgen dat de node `ssn` een geldig nummer bevat of dat voor de werknemer ten minste één telefoonnummer en e-mailadres is gedefinieerd en dat beide waarden geldig zijn. U kunt de XML ook zo aanpassen dat elke employee een unieke id heeft en dat daarbij de id van zijn of haar manager is opgegeven.

De klasse ApplicationError definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse ApplicationError fungeert als basisklasse voor de klassen FatalError en WarningError. De klasse ApplicationError breidt de klasse Error uit en definieert eigen aangepaste methoden en eigenschappen, waaronder een fout-id, de ernst en een XML-object met daarin de aangepaste foutcodes en foutmeldingen. Deze klasse definieert ook twee statische constanten waarmee de ernst van elk fouttype wordt gedefinieerd.

De constructormethode van de klasse ApplicationError ziet er als volgt uit:

```
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>;
}
```

Elke foutnode in het XML-object bevat een unieke numerieke code en een foutmelding. Foutmeldingen kunnen gemakkelijk worden opgezocht aan de hand van hun foutcode in E4X. Dit wordt in de volgende methode `getMessageText()` getoond:

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

De methode `getMessageText()` werkt met één argument van een geheel getal (`id`) en retourneert een tekenreeks. Het argument `id` is de foutcode voor de fout die moet worden opgezocht. Wanneer bijvoorbeeld een `id` van 9001 wordt doorgegeven, komt u bij de foutmelding te weten dat medewerkers maar aan één kostenplaats mogen worden toegewezen. Als er meerdere fouten met dezelfde foutcode zijn, retourneert ActionScript alleen de foutmelding van het eerste gevonden resultaat (`message[0]` in het geretourneerde XMLList-object).

De volgende methode in deze klasse (`getTitle()`) heeft geen parameters en retourneert een tekenreekswaarde met daarin de fout-id voor deze specifieke fout. Deze waarde wordt gebruikt om duidelijk te maken welke fout precies is opgetreden tijdens de validatie van het XML-pakket. Het volgende fragment toont de methode `getTitle()`:

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

De laatste methode in de klasse ApplicationError is `toString()`. Deze methode negeert de functie die in de klasse Error is gedefinieerd, waardoor u de presentatie van de foutmelding kunt aanpassen. De methode retourneert een tekenreeks die het specifieke foutnummer en de melding duidelijk maakt.

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

De klasse FatalError definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse FatalError breidt de aangepaste klasse ApplicationError uit en definieert drie methoden: de constructor FatalError, `getTitle()` en `toString()`. De eerste methode, de constructor FatalError, werkt met één argument van een geheel getal (`errorID`) en bepaalt de ernst van de fout met behulp van de waarden van de statische constanten die in de klasse ApplicationError zijn gedefinieerd. De foutmelding van de specifieke fout wordt opgevraagd door de methode `getMessageText()` uit de klasse ApplicationError aan te roepen. De constructor FatalError ziet er als volgt uit:

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

De volgende methode in de klasse FatalError (`getTitle()`) negeert de methode `getTitle()` die eerder in de klasse ApplicationError is gedefinieerd en voegt de tekst '-- FATAL' toe aan de titel om de gebruiker duidelijk te maken dat een fatale fout is opgetreden. De methode `getTitle()` ziet er als volgt uit:

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

De laatste methode in deze klasse (`toString()`) negeert de methode `toString()` die in de klasse ApplicationError is gedefinieerd. De methode `toString()` ziet er als volgt uit:

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

De klasse WarningError definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse WarningError breidt de klasse ApplicationError uit en is vrijwel identiek aan de klasse FatalError op een paar kleine wijzigingen in de tekenreeksen na. Hiermee wordt de ernst van de fout op ApplicationError.WARNING in plaats van ApplicationError.FATAL gezet, zoals in de volgende code wordt getoond:

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

Hoofdstuk 5: Reguliere expressies gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een reguliere expressie beschrijft een patroon dat wordt gebruikt om te zoeken naar identieke tekst in tekenreeksen en deze te bewerken. Reguliere expressies bootsen tekenreeksen na, maar kunnen speciale codes bevatten voor het beschrijven van patronen en herhalingen. De volgende reguliere expressie komt bijvoorbeeld overeen met een tekenreeks die begint met het teken A en een of meer opeenvolgende cijfers:

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

De volgende onderwerpen beschrijven de basissyntaxis voor het maken van reguliere expressies. Reguliere expressies kunnen echter vele complexe onderdelen en nuances bevatten. U kunt gedetailleerde informatie over reguliere expressies vinden op internet en in uw boekwinkel. De daadwerkelijke implementatie van reguliere expressies hangt af van de programmeeromgeving. ActionScript 3.0 implementeert reguliere expressies zoals gedefinieerd in de ECMAScript Language Specification (ECMA-262), 3rd Edition.

Meer Help-onderwerpen

RegExp

Basisbeginselen van reguliere expressies

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een reguliere expressie beschrijft een patroon van tekens. Reguliere expressies worden doorgaans gebruikt om te bevestigen dat een tekstwaarde voldoet aan een specifiek patroon (bijvoorbeeld om te bevestigen dat een telefoonnummer het juiste aantal cijfers bevat) of om delen van een tekstwaarde te vervangen die overeenkomen met een specifiek patroon.

Reguliere expressies kunnen eenvoudig zijn. Stel dat u bijvoorbeeld wilt bevestigen dat een specifieke tekenreeks overeenkomt met 'ABC', of dat u elke 'ABC' die in een tekenreeks wordt gevonden, wilt vervangen door andere tekst. In dat geval kunt u gebruikmaken van de volgende reguliere expressie, die het patroon definieert dat achtereenvolgens uit de letters A, B en C bestaat:

```
/ABC/
```

Het letterlijke gedeelte van de reguliere expressie wordt afgebakend door de slash (`/`).

Reguliere-expressiepatronen kunnen ook complex en cryptisch overkomen, zoals de volgende expressie voor een geldig e-mailadres:

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

Meestal zult u reguliere expressies gebruiken om te zoeken naar patronen in tekenreeksen en om tekens te vervangen. In dat geval zult u een reguliere-expressieobject maken en dit gebruiken als parameter voor een methode voor de klasse String. Bij de volgende methoden van de klasse String worden reguliere expressies als parameter gebruikt: `match()`, `replace()`, `search()` en `split()`. Zie "Patronen in tekenreeksen zoeken en subtekenreeksen vervangen" op pagina 17 voor meer informatie over deze methoden.

De klasse RegExp bevat de volgende methoden: `test()` en `exec()`. Zie "Methoden voor het gebruik van reguliere expressies met tekenreeksen" op pagina 95 voor meer informatie.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen die relevant zijn voor deze functie:

Escape-teken Een teken dat aangeeft dat het volgteken moet worden behandeld als een metateken in plaats van een letterlijk teken. In de syntaxis van reguliere expressies is de backslash (\) het escape-teken; daarom stelt een backslash gevolgd door een ander teken een speciale code voor in plaats van het teken zelf.

Markering Een teken dat een optie aangeeft over de wijze waarop het reguliere-expressiepatroon moet worden gebruikt, bijvoorbeeld of er onderscheid moet worden gemaakt tussen hoofdletters en kleine letters.

Metateken Een teken dat een speciale betekenis heeft in een reguliere-expressiepatroon en dus niet het teken in het patroon letterlijk vertegenwoordigt.

Kwantor Een teken (of meerdere tekens) dat aangeeft hoe vaak een deel van het patroon moet worden herhaald. Een kwantor kan bijvoorbeeld worden gebruikt om aan te geven dat postcodes in de Verenigde Staten uit vijf of negen cijfers moeten bestaan.

Reguliere expressie Een programma-instructie voor het definiëren van een patroon van tekens. Deze instructie kan worden gebruikt om te bevestigen of andere tekenreeksen overeenkomen met dat patroon, of om delen van een tekenreeks te vervangen.

Syntaxis voor reguliere expressies

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In deze sectie worden alle elementen van de syntaxis van reguliere expressies in ActionScript beschreven. Zoals u zult zien, kunnen reguliere expressies vele complexe onderdelen en nuances bevatten. U kunt gedetailleerde informatie over reguliere expressies vinden op internet en in uw boekwinkel. De daadwerkelijke implementatie van reguliere expressies hangt af van de programmeeromgeving. ActionScript 3.0 implementeert reguliere expressies zoals gedefinieerd in de ECMAScript Language Specification (ECMA-262), 3rd Edition.

Gewoonlijk gebruikt u reguliere expressies die overeenkomen met meer gecompliceerde patronen dan een eenvoudige tekenreeks. De volgende reguliere expressie definieert bijvoorbeeld het patroon bestaande uit de letters A, B en C gevolgd door een willekeurig cijfer:

```
/ABC\d/
```

De code `\d` vertegenwoordigt een 'willekeurig cijfer'. De backslash (`\`) wordt het escape-teken genoemd. In combinatie met het daaropvolgende teken (hier de letter d), heeft het een speciale betekenis in de reguliere expressie.

De volgende reguliere expressie definieert het patroon van de letters ABC gevolgd door een willekeurig aantal cijfers (houd rekening met de asterisk):

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

De asterisk (`*`) is een *metateken*. Een metateken is een teken met een speciale betekenis in reguliere expressies. De asterisk is een specifiek metateken dat *kwantor* heet en aangeeft hoe vaak een teken of groep tekens voorkomt. Zie "Kwantoren" op pagina 88 voor meer informatie.

Een reguliere expressie kan behalve zijn patroon ook markeringen bevatten, die aangeven hoe de reguliere expressie moet overeenkomen. De volgende reguliere expressie bijvoorbeeld gebruikt de markering `i`, die aangeeft dat de reguliere expressie in overeenkomstige tekenreeksen hoofdlettergevoeligheid negeert:

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

Zie "Markeringen en eigenschappen" op pagina 92 voor meer informatie.

U kunt reguliere expressies met de volgende methoden van de klasse String gebruiken: `match()`, `replace()` en `search()`. Zie "Patronen in tekenreeksen zoeken en subtekenreeksen vervangen" op pagina 17 voor meer informatie over deze methoden.

Instantie van een reguliere expressie maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een reguliere-expressie-instantie kan op twee manieren worden gemaakt. Bij de eerste manier worden slash-tekens (`/`) gebruikt om de reguliere expressie af te bakenen; bij de andere manier wordt de constructor `new` gebruikt. De volgende twee reguliere expressies zijn bijvoorbeeld gelijk:

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

Slash-tekens bakenen een reguliere expressie op dezelfde manier af als aanhalingstekens een tekenreeks letterlijk afbakenen. Het deel van de reguliere expressie binnen de slashes definieert het *patroon*. De reguliere expressie kan tevens *markeringen* na de afsluitende afbakeningsslash bevatten. Deze markeringen worden beschouwd als deel van de reguliere expressie, maar staan los van het patroon ervan.

Wanneer u de constructor `new` gebruikt, definieert u de reguliere expressie met behulp van twee tekenreeksen. De eerste tekenreeks definieert het patroon, de tweede tekenreeks definieert de markeringen, zoals in het volgende voorbeeld:

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

Wanneer er zich een slash *in* een reguliere expressie bevindt die is gedefinieerd met behulp van de slash-afbakeningstekens, laat u de slash voorafgaan door het escape-teken, de backslash (`\`). De volgende reguliere expressie komt bijvoorbeeld overeen met het patroon `1/2`:

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

Om aanhalingstekens *binnen* een reguliere expressie op te nemen die is gedefinieerd met de constructor `new`, moet u een backslash (`\`) als escape-teken toevoegen vóór de aanhalingstekens (zoals u zou doen bij het letterlijk definiëren van een willekeurige tekenreeks). De volgende reguliere expressies komen bijvoorbeeld overeen met het patroon `eat at "joe's"`:

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

Gebruik de backslash niet als escape-teken met aanhalingstekens in reguliere expressies die zijn gedefinieerd door middel van de slash-afbakeningstekens. Gebruik het escape-teken met slashes evenmin in reguliere expressies die gedefinieerd zijn met de constructor `new`. De volgende reguliere expressies zijn gelijk en definiëren het patroon `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"', '');
```

Wanneer u in een reguliere expressie die is gedefinieerd met de constructor `new` een metareeks wilt gebruiken die met een backslash (`\`) begint, zoals `\d` (wat staat voor elk willekeurig cijfer), typt u de backslash twee keer:

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

U moet het backslash-teken in dit geval twee keer typen. De eerste parameter van de constructormethode `RegExp()` is namelijk een tekenreeks, en in een letterlijke tekenreeks moet een backslash-teken twee keer worden getypt om als enkel backslash-teken te worden herkend.

In de volgende secties wordt de syntaxis voor het definiëren van reguliere-expressiepatronen beschreven.

Zie "Markeringen en eigenschappen" op pagina 92 voor meer informatie over markeringen.

Tekens, metatekens en metareeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eenvoudigste reguliere expressie komt overeen met een reeks tekens, zoals in het volgende voorbeeld:

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

De volgende tekens echter, die metatekens worden genoemd, hebben een speciale betekenis in reguliere expressies:

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

De volgende reguliere expressie bijvoorbeeld komt overeen met de letter A gevolgd door nul of meer instanties van de letter B (de asterisk geeft deze herhaling aan), gevolgd door de letter C:

```
/AB*C/
```

Als u een metateken in een reguliere-expressiepatroon wilt opnemen zonder de speciale betekenis, moet u de backslash (`\`) als escape-teken gebruiken. De volgende reguliere expressie komt bijvoorbeeld overeen met de letter A gevolgd door de letter B, gevolgd door een asterisk, gevolgd door de letter C:

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

Een *metareeks* heeft, net als een metateken, een speciale betekenis in een reguliere expressie. Een metareeks bestaat uit meer dan een teken. De volgende secties geven informatie over het gebruik van metatekens en metareeksen.

Informatie over metatekens

De volgende tabel geeft een overzicht van de metatekens die u kunt gebruiken in reguliere expressies:

Metateken	Beschrijving
^ (invoegpunt)	Komt overeen met het begin van de tekenreeks. Als de markering `m` (`meerdere regels`) is ingesteld, komt de invoegpunt ook overeen met het begin van een regel (zie "Markeringen en eigenschappen" op pagina 92). Wanneer de invoegpunt wordt gebruikt aan het begin van een tekenklasse, geeft de invoegpunt een negatie aan, niet het begin van een tekenreeks. Zie "Tekenklassen" op pagina 86 voor meer informatie.
$(dollarteken)	Komt overeen met het einde van de tekenreeks. Als de markering `m` (`meerdere regels`) is ingesteld, komt `$` ook overeen met de positie vóór het teken (`\n`) voor een nieuwe regel. Zie "Markeringen en eigenschappen" op pagina 92 voor meer informatie.
\ (backslash)	Verwijdert de speciale betekenis van metatekens. Gebruik het backslash-teken ook als u een slash-teken letterlijk wilt gebruiken in een reguliere expressie, zoals in `/1\/2/` (om overeen te komen met teken 1, gevolgd door een slash, gevolgd door het teken 2).

Metateken	Beschrijving
. (punt)	Komt overeen met elk willekeurig enkel teken. Een punt komt alleen overeen met een nieuwe-regelteken (`\n`) als de markering`s` (`dotall`) in ingesteld. Zie "Markeringen en eigenschappen" op pagina 92 voor meer informatie.
* (ster)	Komt overeen met het vorige item dat nul keer of vaker is herhaald. Zie "Kwantoren" op pagina 88 voor meer informatie.
+ (plus)	Komt overeen met het vorige item dat een keer of vaker is herhaald. Zie "Kwantoren" op pagina 88 voor meer informatie.
? (vraagteken)	Komt overeen met het vorige item dat nul of een keer is herhaald. Zie "Kwantoren" op pagina 88 voor meer informatie.
(en)	Definieert groepen binnen de reguliere expressie. Gebruik groepen voor de volgende handelingen: • Het bereik beperken van de verticale balk (|): `/(a|b|c)d/` • Het bereik bepalen van een kwantor: `/(walla.){1,2}/` • In terugverwijzingen. De `\1` in de volgende reguliere expressie komt bijvoorbeeld overeen met alles uit de eerste overeenkomende groep tussen haakjes van het patroon: • `/(\w*) is herhaald: \1/` Zie "Groepen" op pagina 89 voor meer informatie.
[en]	Definieert een tekenklasse, die mogelijke overeenkomsten met een afzonderlijk teken definieert: `/[aeiou]/` komt overeen met een van de opgegeven tekens. Gebruik binnen tekenklassen het koppelteken (-) om een reeks tekens op te geven: `/[A-Z0-9]/` komt overeen met A t/m Z in hoofdletters of 0 t/m 9. Voeg binnen tekenklassen een backslash in om de tekens] en - te beschermen: `/[+\-]\d+/` komt ofwel overeen met + ofwel - vóór een of meer cijfers. Binnen andere tekenklassen worden andere tekens, normaal gesproken metatekens, behandeld als normale tekens (en niet als metatekens), zonder dat er een backslash moet worden gebruikt: `/[$]/`£ komt ofwel met `$of` £ overeen. Zie "Tekenklassen" op pagina 86 voor meer informatie.
| *(verticale balk)*	Wordt gebruikt voor een alternatief, om ofwel met het deel links ofwel met het deel rechts overeen te komen: `/abc|xyz/` komt ofwel met `abc` ofwel met `xyz` overeen.

Informatie over metareeksen

Metareeksen zijn reeksen tekens met een speciale betekenis binnen een reguliere-expressiepatroon. In de volgende tabel worden deze metareeksen beschreven.

Metareeks	Beschrijving
{*n*} {*znw*,} and {*n*,*n*}	Geeft een numerieke kwantor of een numeriek kwantorbereik voor het vorige item aan: `/A{27}/` komt overeen met het teken `A` dat `27` keer wordt herhaald. `/A{3,}/` komt overeen met het teken `A` dat `3` of meer keer wordt herhaald. `/A{3,5}/` komt overeen met het teken `A` dat `3` tot `5` keer wordt herhaald. Zie "Kwantoren" op pagina 88 voor meer informatie.
`\b`	Komt overeen met de positie tussen een woordteken en een teken anders dan een woord. Als het eerste of het laatste teken binnen de tekenreeks een woordteken is, komt het ook overeen met het begin of einde van de tekenreeks.
`\B`	Komt overeen met de positie tussen twee woordtekens. Komt ook overeen met de positie tussen twee tekens anders dan voor woorden.
`\d`	Komt overeen met een decimaal cijfer.
`\D`	Komt overeen met elk teken anders dan een cijfer.
`\f`	Komt overeen met een paginadoorvoerteken.
`\n`	Komt overeen met het nieuwe-regelteken.
`\r`	Komt overeen met het Enter-teken.
`\s`	Komt overeen met een willekeurige witruimte (een spatie, tab, nieuwe-regelteken, of Enter-teken).
`\S`	Komt overeen met ieder teken anders dan voor witruimte.
`\t`	Komt overeen met het tab-teken.
`\u`*nnnn*	Komt overeen met het Unicode-teken met de tekencode aangeduid door het hexadecimale getal *nnnn*. `\u263a` is bijvoorbeeld het smiley-teken.
`\v`	Komt overeen met een verticaal doorvoerteken.
`\w`	Komt overeen met een woordteken (`A`–`Z`, `a`–`z`, `0`–`9` of `_`). De reeks `\w` komt niet overeen met tekens met accenten, zoals `é`, `ñ` of `ç`.
`\W`	Komt overeen met ieder ander teken dan een woordteken.
`\\x`*nn*	Komt overeen met de aangegeven ASCII-waarde, zoals gedefinieerd door het hexadecimale getal *nn*.

Tekenklassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Tekenklassen gebruikt u om een lijst op te geven met tekens die overeen moeten komen met een positie in de reguliere expressie. Tekenklassen definieert u door middel van vierkante haakjes (`[` en `]`). De volgende reguliere expressie definieert bijvoorbeeld een tekenklasse die overeenkomt met `bag`, `beg`, `big`, `bog` of `bug`:

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

Escape-reeksen in tekenklassen

De meeste metatekens en metareeksen, die normaal gesproken een speciale betekenis hebben binnen een reguliere expressie, *hebben niet* dezelfde betekenis binnen een tekenklasse. In een reguliere expressie bijvoorbeeld wordt de asterisk gebruikt voor herhalingen; dit is echter niet het geval wanneer de asterisk binnen een tekenklasse verschijnt. De volgende tekenklasse komt letterlijk overeen met de asterisk, samen met de andere opgenomen tekens:

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

De drie tekens echter die zijn opgenomen in de volgende tabel, functioneren als metatekens met een speciale betekenis in tekenklassen:

Metateken	Betekenis in tekenklassen
`]`	Definieert het einde van de tekenklasse.
`-`	Hiermee wordt een tekenbereik gedefinieerd (zie de volgende sectie, Tekenbereiken binnen tekenklassen).
`\`	Hiermee worden metareeksen gedefinieerd en wordt de speciale betekenis van metatekens ongedaan gemaakt.

Om ervoor te zorgen dat deze tekens worden herkend als letterlijke tekens (zonder de speciale betekenis behorend bij het metateken), laat u het teken voorafgaan door een backslash. De volgende reguliere expressie bijvoorbeeld bevat een tekenklasse die overeenkomt met elk van de volgende vier symbolen (`$`, `\`, `]` of `-`):

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

Naast de metatekens, die hun speciale betekenis behouden, functioneren de volgende metareeksen als metareeksen binnen tekenklassen:

Metareeks	Betekenis in tekenklassen
`\n`	Komt overeen met een nieuwe-regelteken.
`\r`	Komt overeen met een Return-teken.
`\t`	Komt overeen met een tab-teken.
`\u`***nnnn***	Komt overeen met het Unicode-teken met de tekencode aangeduid door het hexadecimale getal *nnnn*).
`\\x`***nn***	Komt overeen met de ASCII-waarde aangeduid door het hexadecimale getal *nn*).

Andere metareeksen en metatekens binnen reguliere expressies worden behandeld als normale tekens binnen een tekenklasse.

Tekenbereiken binnen tekenklassen

Gebruik het koppelteken om een tekenbereik aan te geven, zoals `A-Z`, `a-z` of `0-9`. Deze tekens moeten een geldig bereik binnen de tekenset vormen. De volgende tekenklasse komt bijvoorbeeld overeen met elk van de tekens binnen het bereik `a-z` of elk willekeurig cijfer:

```
/[a-z0-9]/
```

U kunt ook de ASCII-tekencode `\\x`*nn* gebruiken om een bereik met ASCII-waarde op te geven. De volgende tekenklasse komt overeen met elk teken uit een uitgebreide set ASCII-tekens (zoals é en ê):

```
\\x
```

Genegeerde tekenklassen

Wanneer u een invoegpunt gebruikt (`^`) aan het begin van een tekenklasse, wordt deze klasse door dit teken genegeerd: elk willekeurig teken dat niet is opgenomen, wordt als overeenkomend beschouwd. De volgende tekenklasse komt overeen met elk willekeurig teken *behalve* voor een kleine letter (`az`–) of een cijfer:

```
/[^a-z0-9]/
```

Typ het invoegteken (`^`) aan het *begin* van een tekenklasse om de negatie aan te geven. Anders voegt u het invoegteken eenvoudig toe aan de tekens in de tekenklasse. De volgende tekenklasse bijvoorbeeld komt overeen met elk willekeurig aantal symbooltekens, waaronder de invoegpunt:

```
/[!.,#+*%$&^]/
```

Kwantoren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruik kwantoren om herhalingen van tekens of reeksen in patronen als volgt aan te geven:

Metateken voor kwantor	Beschrijving
`*` (ster)	Komt overeen met het vorige item dat nul keer of vaker is herhaald.
`+` (plus)	Komt overeen met het vorige item dat een keer of vaker is herhaald.
`?` (vraagteken)	Komt overeen met het vorige item dat nul of een keer is herhaald.
{*n*} {*znw*,} and {*n*,*n*}	Geeft een numerieke kwantor of een numeriek kwantorbereik voor het vorige item aan: `/A{27}/` komt overeen met het teken A dat 27 keer wordt herhaald. `/A{3,}/` komt overeen met het teken A 3 of meer keer wordt herhaald. `/A{3,5}/` komt overeen met het teken A dat 3 tot 5 keer wordt herhaald.

U kunt een kwantor toevoegen aan een afzonderlijk teken, aan een tekenklasse of aan een groep:

- `/a+/` komt overeen met het teken `a` dat een of meer keer wordt herhaald.
- `/\d+/` komt overeen met een of meer cijfers.
- `/[abc]+/` komt overeen met een herhaling van een of meer tekens, waarvan elk ofwel `a`, `b` of `c` is.
- `/(very, )*/` komt overeen met het woord `very`, gevolgd door een komma en een spatie, dat nul keer of vaker wordt herhaald.

U kunt kwantoren binnen overeenkomende groepen tussen haakjes gebruiken waaraan kwantoren zijn toegevoegd. De volgende kwantor komt bijvoorbeeld overeen met tekenreeksen zoals `word` en `word-word-word`:

```
/\w+(-\w+)*/
```

Reguliere expressies zorgen voor wat *greedy matching wordt genoemd.* Elk subpatroon binnen de reguliere expressie (zoals `.*`) probeert met zoveel mogelijk tekens in de tekenreeks overeen te komen voordat naar het volgende gedeelte van de reguliere expressie wordt gegaan. Kijk bijvoorbeeld naar de volgende reguliere expressie en tekenreeks:

```
var pattern:RegExp = /<p>.*<\/p>/;
str:String = "<p>Paragraph 1</p> <p>Paragraph 2</p>";
```

De reguliere expressie komt overeen met de volledige tekenreeks:

```
<p>Paragraph 1</p> <p>Paragraph 2</p>
```

Stel echter dat er slechts één serie `<p>...</p>` moet overeenkomen. Dit kunt u doen op de volgende manier:

```
<p>Paragraph 1</p>
```

Voeg een vraagteken toe (`?`) na een willekeurige kwantor, om deze te wijzigen in een zogenaamde *lazy kwantor*. De volgende reguliere expressie bijvoorbeeld, die gebruikmaakt van de lazy kwantor `*?`, komt overeen met `<p>` gevolgd door het minimaal mogelijke aantal tekens (lazy), gevolgd door `</p>`:

```
/<p>.*?<\/p>/
```

Houd bij kwantoren rekening met de volgende punten:

- De kwantoren `{0}` en `{0,0}` sluiten geen item uit als er een overeenkomst is.

- Maak geen combinaties van meerdere kwantoren, zoals in `/abc+*/`.
- De punt (.) zorgt niet voor een bereik van regels, tenzij de markering `s` (`dotall`) is ingesteld, zelfs als deze wordt gevolgd door een kwantor `*`. Neem bijvoorbeeld de volgende code:

```
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>
```

Zie "Markeringen en eigenschappen" op pagina 92 voor meer informatie.

Alternatieven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u het teken `|` (verticale balk) in een reguliere expressie gebruikt, zoekt de engine voor de reguliere expressie naar alternatieven voor een overeenkomst. De volgende reguliere expressie bijvoorbeeld komt overeen met elk van de woorden `cat, dog, pig, rat`:

```
var pattern:RegExp = /cat|dog|pig|rat/;
```

U kunt ronde haken gebruiken om groepen te definiëren waarmee u het bereik van de verticale balk `|` beperkt. De volgende reguliere expressie komt overeen met `cat` gevolgd door `nap` of `nip`:

```
var pattern:RegExp = /cat(nap|nip)/;
```

Zie "Groepen" op pagina 89 voor meer informatie.

De volgende twee reguliere expressies, waarvan de ene de verticale balk `|` en de andere een tekenklasse gebruikt (gedefinieerd met `[` en `]`), zijn gelijk:

```
/1|3|5|7|9/
/[13579]/
```

Zie "Tekenklassen" op pagina 86 voor meer informatie.

Groepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een groep als volgt aangeven binnen een reguliere expressie met behulp van ronde haken:

```
/class-(\d*)/
```

Een groep is een subsectie van een patroon. U kunt groepen gebruiken om de volgende handelingen uit te voeren:

- Een kwantor toevoegen aan meer dan een teken.
- Subpatronen afbakenen om toe te passen met alternatieven (door gebruik te maken van het teken `|`).
- Overeenkomende subtekenreeksen vastleggen (bijvoorbeeld door het gebruik van `\1` in een reguliere expressie met een eerder overeenkomende groep, of door gebruik te maken van `$1` zoals in methode `replace()` van de klasse String).

De volgende secties geven informatie over het gebruik van deze groepen.

Groepen gebruiken met kwantoren

Als u geen groep gebruikt, heeft een kwantor betrekking op het teken of de tekenklasse die hieraan voorafgaat, zoals uit het onderstaande blijkt:

```
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
```

U kunt echter een groep gebruiken om een kwantor toe te voegen aan meer dan een teken of tekenklasse:

```
var pattern:RegExp = /(ab)*/;
// matches zero or more occurrences of the character a
// followed by the character b, such as abababab

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
```

Zie “Kwantoren” op pagina 88 voor meer informatie over kwantoren.

Groepen gebruiken met de verticale balk (|)

U kunt groepen gebruiken om de groep tekens te definiëren waarop u een verticale balk (|) wilt toepassen:

```
var pattern:RegExp = /cat|dog/;
// matches cat or dog

pattern = /ca(t|d)og/;
// matches catog or cadog
```

Groepen gebruiken om overeenkomstige subreeksen vast te leggen

Wanneer u een standaardgroep tussen haakjes in een patroon vastlegt, kunt u er later in de reguliere expressie naar verwijzen. Dit wordt *terugverwijzing* genoemd, en deze soort groepen worden *vastgelegde groepen* genoemd In de volgende reguliere expressie bijvoorbeeld komt de reeks `\1` overeen met een willekeurige subreeks die met de vastgelegde groep tussen haakjes overeenkomt:

```
var pattern:RegExp = /(\d+)-by-\1/;
// matches the following: 48-by-48
```

U kunt maximaal 99 van deze terugverwijzingen vastleggen in een reguliere expressie door `\1`, `\2`, ..., `\99` te typen.

Op dezelfde manier kunt u in de methode `replace()` van de klasse String `$1$99`– gebruiken om overeenkomende vastgelegde groepsubreeksen in de vervangende tekenreeks in te voegen:

```
var pattern:RegExp = /Hi, (\w+)\./;
var str:String = "Hi, Bob.";
trace(str.replace(pattern, "$1, hello."));
    // output: Bob, hello.
```

Als u vastgelegde groepen gebruikt, retourneren de methode `exec()` van de klasse RegExp en de methode `match()` van de klasse String subreeksen die overeenkomen met de vastgelegde groepen:

```
var pattern:RegExp = /(\w+)@(\w+).(\w+)/;
var str:String = "bob@example.com";
trace(pattern.exec(str));
    // bob@example.com,bob,example,com
```

Gebruiken van niet vastgelegde groepen en lookahead-groepen

Een niet vastgelegde groep is een groep die alleen wordt gebruikt om mee te groeperen; deze wordt niet gebruikt om mee te 'verzamelen' en bevat geen overeenkomsten met genummerde terugverwijzingen. Gebruik `(?:` en `)` om niet vastgelegde groepen als volgt te definiëren:

```
var pattern = /(?:com|org|net);
```

Houd bijvoorbeeld rekening met het verschil wanneer `(com|org)` in een vastgelegde versus een niet vastgelegde groep wordt geplaatst (de methode `exec()` vormt een opsomming van vastgelegde groepen na volledige overeenkomst):

```
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
```

Een bijzonder soort niet vastgelegde groep is de groep *lookahead* waarvan twee types bestaan: de *positieve lookahead-groep* en de *negatieve lookahead-groep.*

Gebruik `(?=` en `)` om een positieve lookahead-groep te definiëren, waarmee wordt aangegeven dat het subpatroon in de groep moet overeenkomen met de positie. Het gedeelte van de tekenreeks dat echter overeenkomt met de positieve lookahead-groep, kan overeenkomen met overige patronen binnen de reguliere expressie. Aangezien `(?=e)` bijvoorbeeld een positieve lookahead-groep binnen de volgende code is, kan het teken `e` waarmee het overeenkomt in overeenstemming worden gebracht met een volgend deel van de reguliere expressie, in dit geval de vastgelegde groep, `\w*)`:

```
var pattern:RegExp = /sh(?=e)(\w*)/i;
var str:String = "Shelly sells seashells by the seashore";
trace(pattern.exec(str));
// Shelly,elly
```

Gebruik `(?!` en `)` om een negatieve lookahead-groep te definiëren die aangeeft dat het subpatroon in de groep *niet* moet overeenkomen met de positie. Bijvoorbeeld:

```
var pattern:RegExp = /sh(?!e)(\w*)/i;
var str:String = "She sells seashells by the seashore";
trace(pattern.exec(str));
// shore,ore
```

Gebruiken van benoemde groepen

Een benoemde groep is een soort groep in een reguliere expressie waaraan een benoemde id is toegewezen. Gebruik `(?P<name>` en `)` om de benoemde groep te definiëren. De volgende reguliere expressie bevat bijvoorbeeld een groep met een benoemde id `digits`:

```
var pattern = /[a-z]+(?P<digits>\d+)[a-z]+/;
```

Wanneer u de methode `exec()` gebruikt, wordt een overeenkomende benoemde groep toegevoegd als eigenschap van de array `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
```

Hier volgt nog een voorbeeld, waarbij gebruik wordt gemaakt van twee benoemde groepen, met de id's `name` en `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
```

Opmerking: *Benoemde groepen maken geen deel uit van de taalspecificatie ECMAScript. Deze vormen een toegevoegde functie binnen ActionScript 3.0.*

Markeringen en eigenschappen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De volgende tabel toont de vijf markeringen die u kunt instellen voor reguliere expressies. Elke markering kan worden benaderd als eigenschap van het reguliere-expressieobject.

Markering	Eigenschap	Beschrijving
`g`	`global`	Komt meer dan een keer overeen.
`i`	`ignoreCase`	Niet-hoofdlettergevoelige overeenkomsten. Heeft betrekking op de tekens `A-Z` en `a-z`, maar niet op uitgebreide tekens zoals `É` en `é`.
`m`	`meerdere regels`	Wanneer deze markering is ingesteld, kunnen `$` en `^` overeenkomen met respectievelijk het begin en einde van een regel.
`s`	`dotall`	Wanneer deze markering is ingesteld, `.` kan (dot) overeenkomen met het teken voor de nieuwe regel (`\n`).
`x`	`uitgebreid`	Hiermee worden uitgebreide reguliere expressies toegestaan. U kunt spaties in de reguliere expressie typen, die genegeerd worden als deel van het patroon. Hiermee kunt u reguliere-expressiecode leesbaarder typen.

Deze eigenschappen zijn alleen-lezen. U kunt de markeringen (`g`, `i`, `m`, `s`, `x`) als volgt instellen wanneer u een reguliere-expressievariabele instelt:

```
var re:RegExp = /abc/gimsx;
```

U kunt de benoemde eigenschappen echter niet rechtstreeks instellen. De volgende code resulteert bijvoorbeeld in een fout:

```
var re:RegExp = /abc/;
re.global = true; // This generates an error.
```

De markeringen worden standaard niet ingesteld, behalve als u deze in de reguliere-expressiedeclaratie aangeeft, en de overeenkomstige eigenschappen worden eveneens op `false` ingesteld.

Een reguliere expressie kan nog twee andere eigenschappen bevatten:

- De eigenschap `lastIndex` geeft de indexpositie in de tekenreeks aan die moet worden gebruikt voor de volgende aanroep van de methode `exec()` of `test()` van een reguliere expressie.
- De eigenschap `source` geeft de tekenreeks aan die het patroongedeelte van de reguliere expressie definieert.

De markering g (global)

Wanneer de markering `g` (`global`) *niet* is opgenomen, komt een reguliere expressie niet meer dan een keer overeen. Wanneer de markering `g` bijvoorbeeld niet is opgenomen in de reguliere expressie, retourneert de methode `String.match()` slechts een overeenkomende subtekenreeks:

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/;
trace(str.match(pattern)) // output: she
```

Wanneer de markering `g` is ingesteld, retourneert de methode `String.match()` meerdere overeenkomsten, als volgt:

```
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
```

De markering i (ignoreCase)

Standaard zijn reguliere-expressieovereenkomsten hoofdlettergevoelig. Wanneer u de markering `i` (`ignoreCase`) instelt, wordt de hoofdlettergevoeligheid genegeerd. De kleine letter `s` in de reguliere expressie komt bijvoorbeeld niet overeen met de hoofdletter `S`, het eerste teken in de tekenreeks:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/)); // output: 13 -- Not the first character
```

Wanneer echter de markering `i` is ingesteld, komt de reguliere expressie niet overeen met de hoofdletter `S`:

```
var str:String = "She sells seashells by the seashore.";
trace(str.search(/sh/i)); // output: 0
```

De markering `i` negeert hoofdlettergevoeligheid alleen voor de tekens `A–Z` en `a–z`, maar niet voor uitgebreide tekens, zoals `É` en `é`.

De markering m (multiline)

Als de markering `m` (`meerdere regels`) niet is ingesteld, komt `^` overeen met het begin van de tekenreeks en `$` met het einde van de tekenreeks. Als de markering `m` is ingesteld, komen deze tekens overeen met het begin van een regel en het einde van een regel. Bekijk de volgende tekenreeks, die het teken voor een nieuwe regel bevat:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/g)); // Match a word at the beginning of the string.
```

Ook al is de markering `g` (`global`) in de reguliere expressie ingesteld, komt de methode `match()` met slechts een subtekenreeks overeen, omdat er slechts een overeenkomst voor `^` is, namelijk het begin van de tekenreeks. De uitvoer is:

```
Test
```

Hier is dezelfde code als de markering `m` is ingesteld:

```
var str:String = "Test\n";
str += "Multiline";
trace(str.match(/^\w*/gm)); // Match a word at the beginning of lines.
```

Nu bevat de uitvoer de woorden aan het begin van beide regels:

```
Test,Multiline
```

Het teken `\n` geeft het einde van een regel aan. De volgende tekens doen dit niet:

- Return-teken (`\r`)
- Unicode-teken voor regelscheiding (`\u2028`)
- Unicode-teken voor alineascheiding (`\u2029`)

De markering s (dotall)

Als de markering `s` (`dotall` of 'dot all') niet is ingesteld, komt een punt (`.`) in een reguliere-expressiepatroon niet overeen met het teken voor een nieuwe regel (`\n`). In het volgende voorbeeld is er daarom geen overeenkomst:

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/;
trace(str.match(re));
```

Als de markering `s` echter is ingesteld, komt de punt overeen met het teken voor de nieuwe regel:

```
var str:String = "<p>Test\n";
str += "Multiline</p>";
var re:RegExp = /<p>.*?<\/p>/s;
trace(str.match(re));
```

In dit geval is de overeenkomst de volledige subtekenreeks binnen de tags `<p>`, waaronder het teken voor de nieuwe regel:

```
<p>Test
Multiline</p>
```

De markering x (uitgebreid)

Reguliere expressies zijn soms moeilijk leesbaar, met name als ze veel speciale symbolen en metareeksen bevatten. Bijvoorbeeld:

```
/<p(>|(\s*[^>]*>)).*?<\/p>/gi
```

Wanneer u de markering `x` (`extended`) in een reguliere expressie gebruikt, worden lege spaties die u binnen het patroon typt, genegeerd. De volgende reguliere expressie is bijvoorbeeld identiek aan het vorige voorbeeld:

```
/     <p    (>  |  (\s*  [^>]*  >))    .*?    <\/p>  /gix
```

Als u de markering `x` hebt ingesteld en u wilt dat een leeg spatieteken overeenkomt, plaatst u voor dit lege spatieteken een backslash. De volgende twee reguliere expressies zijn bijvoorbeeld gelijkwaardig:

```
/foo bar/
/foo \ bar/x
```

De eigenschap lastIndex

De eigenschap `lastIndex` geeft de indexpositie aan in de tekenreeks waar de volgende zoekopdracht moet beginnen. Deze eigenschap heeft invloed op de methoden `exec()` en `test()` die worden aangeroepen in een reguliere expressie waarbij de markering `g` is ingesteld op `true`. Neem bijvoorbeeld de volgende code:

```
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);
}
```

De eigenschap `lastIndex` is standaard ingesteld op 0 (om zoekacties te starten aan het begin van de tekenreeks). Na elke overeenkomst wordt deze ingesteld op de indexpositie volgend op de overeenkomst. Daarom is de uitvoer van de voorgaande code als volgt:

```
0
5
11
18
25
36
44
```

Als de markering `global` is ingesteld op `false`, gebruiken de methoden `exec()` en `test()` niet de eigenschap `lastIndex` of stellen deze de index niet in.

De methoden `match()`, `replace()` en `search()` van de klasse String starten alle zoekacties vanaf het begin van de tekenreeks, ongeacht de instelling van de eigenschap `lastIndex` van de reguliere expressie die gebruikt is in de aanroep van de methode. (De methode `match()` stelt `lastIndex` echter in op 0.)

U kunt de eigenschap `lastIndex` zo instellen dat de startpositie in de tekenreeks voor overeenkomsten van de reguliere expressie wordt aangepast.

De eigenschap source (bron)

De eigenschap `source` geeft de tekenreeks aan die het patroongedeelte van de reguliere expressie definieert. Bijvoorbeeld:

```
var pattern:RegExp = /foo/gi;
trace(pattern.source); // foo
```

Methoden voor het gebruik van reguliere expressies met tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse RegExp bevat twee methoden: `exec()` en `test()`.

Naast de methoden `exec()` en `test()` van de klasse RegExp, bevat de klasse String de volgende methoden waarmee u reguliere expressies in tekenreeksen kunt laten overeenkomen: `match()`, `replace()`, `search()` en `splice()`.

De methode test()

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `test()` uit de klasse RegExp controleert eenvoudig de aangeboden tekenreeks om te zien of deze een overeenkomst bevat voor de reguliere expressie, zoals blijkt uit het volgende voorbeeld:

```
var pattern:RegExp = /Class-\w/;
var str = "Class-A";
trace(pattern.test(str)); // output: true
```

De methode exec()

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `exec()` van de klasse RegExp controleert de aangeboden tekenreeks op een overeenkomst van de reguliere expressie en retourneert een array met de volgende punten:

- De overeenkomende subtekenreeks
- Subtekenreeks komt overeen voor groepen tussen ronde haakjes in de reguliere expressie

De array bevat daarnaast een eigenschap `index`, die de indexpositie van het begin van de overeenkomende subtekenreeks aangeeft.

Neem bijvoorbeeld de volgende code:

```
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
```

Gebruik de methode `exec()` meerdere keren om meerdere subtekenreeksen overeen te laten komen wanneer de markering `g` (`global`) is ingesteld voor de reguliere expressie:

```
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
```

Tekenreeks-methoden die gebruikmaken van RegExp-parameters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij de volgende methoden van de klasse String worden reguliere expressies als parameter gebruikt: `match()`, `replace()`, `search()` en `split()`. Zie "Patronen in tekenreeksen zoeken en subtekenreeksen vervangen" op pagina 17 voor meer informatie over deze methoden.

Voorbeeld van een reguliere expressie: een Wiki-parser

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Dit eenvoudige conversievoorbeeld van Wiki-tekst geeft een aantal toepassingen voor reguliere expressies:

- Het omzetten van tekstregels die overeenkomen met een Wiki-bronpatroon in de juiste tekenreeksen in HTML-indeling.
- Een reguliere expressie gebruiken om URL-patronen in HTML-tags `<a>` voor hyperlinks om te zetten.
- Een reguliere expressie gebruiken om tekenreeksen in Amerikaanse dollar (zoals `"$9.95"`) om te zetten in tekenreeksen met het euroteken (zoals `"8.24 €"`).

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van WikiEditor vindt u in de map Samples/WikiEditor. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
WikiEditor.mxml of WikiEditor.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/regExpExamples/WikiParser.as	Een klasse die methoden bevat die gebruikmaken van reguliere expressies om Wiki-tekstpatronen om te zetten in gelijkwaardige tekst in HTML-indeling.
com/example/programmingas3/regExpExamples/URLParser.as	Een klasse die methoden bevat die gebruikmaken van reguliere expressies om URL-tekenreeksen om te zetten in HTML-tags `<a>` voor hyperlinks.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	Een klasse die methoden bevat die gebruikmaken van reguliere expressies om tekenreeksen met Amerikaanse dollar om te zetten in tekenreeksen met het euroteken.

Klasse WikiParser definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse WikiParser bevat methoden waarmee Wiki-invoertekst wordt omgezet naar gelijkwaardige tekst in HTML-indeling. Dit is geen bijzonder krachtige Wiki-conversietoepassing, maar deze toont wel enkele geschikte toepassingen van reguliere expressies voor patroonovereenkomst en tekenreeksomzetting.

De constructorfunctie initialiseert in combinatie met de methode `setWikiData()` als volgt een voorbeeldtekenreeks van Wiki-invoertekst:

```
public function WikiParser()
{
    wikiData = setWikiData();
}
```

Wanneer de gebruiker klikt op de knop Test in de voorbeeldtoepassing, roept deze de methode `parseWikiString()` van het WikiParser-object aan. Deze methode roept een aantal andere methoden aan, die omgekeerd de tekenreeks in HTML-indeling samenstellen.

```
public function parseWikiString(wikiString:String):String
{
    var result:String = parseBold(wikiString);
    result = parseItalic(result);
    result = linesToParagraphs(result);
    result = parseBullets(result);
    return result;
}
```

Elk van de aangeroepen methoden, `parseBold()`, `parseItalic()`, `linesToParagraphs()` en `parseBullets()`, maakt gebruik van de methode `replace()` van de tekenreeks. Hiermee kunnen overeenkomende patronen, die gedefinieerd zijn door een reguliere expressie, worden vervangen en Wiki-invoertekst vervolgens worden omgezet in tekst in HTML-indeling.

Vetgedrukte en cursieve patronen omzetten

De methode `parseBold()` zoekt naar een vetgedrukt Wiki-tekstpatroon (zoals `'''foo'''`) en zet dit als volgt om in een gelijkwaardig patroon in HTML-indeling (zoals `<b>foo</b>`):

```
private function parseBold(input:String):String
{
    var pattern:RegExp = /'''(.*?)'''/g;
    return input.replace(pattern, "<b>$1</b>");
}
```

Houd er rekening mee dat het gedeelte `(.?*)` van de reguliere expressie overeenkomt met een willekeurig aantal tekens (`*`) tussen de twee gedefinieerde `'''` patronen. De kwantor`?` maakt de overeenkomst nongreedy, zodat voor een tekenreeks als `'''aaa''' bbb '''ccc'''` de eerste overeenkomende tekenreeks `'''aaa'''` is in plaats van de complete tekenreeks (die begint en eindigt met het patroon `'''`).

De ronde haakjes in de reguliere expressie definiëren een vastgelegde groep, en de methode `replace()` verwijst naar deze groep door gebruik te maken van de code `$1` in de vervangende tekenreeks. De markering `g` (`global`) in de reguliere expressie zorgt ervoor dat de methode `replace()` alle overeenkomsten in de tekenreeks (dus niet alleen de eerste) vervangt.

De methode `parseItalic()` werkt hetzelfde als de methode `parseBold()`, behalve dat deze controleert op twee apostroffen (`''`) als scheidingsteken voor cursief gedrukte tekst (in plaats van drie):

```
private function parseItalic(input:String):String
{
    var pattern:RegExp = /''(.*?)''/g;
    return input.replace(pattern, "<i>$1</i>");
}
```

Bullet-patronen omzetten

Zoals uit het onderstaande voorbeeld blijkt, zoekt de methode `parseBullet()` naar Wiki-regelpatroon met bullets (zoals `* foo`) en zet deze in een vergelijkbaar patroon in HTML-indeling (zoals `<li>foo</li>`):

```
private function parseBullets(input:String):String
{
    var pattern:RegExp = /^\*(.*)/gm;
    return input.replace(pattern, "<li>$1</li>");
}
```

Het symbool `^` aan het begin van de reguliere expressie komt overeen met het begin van een regel. De markering `m` (`multiline`) in de reguliere expressie zorgt ervoor dat de reguliere expressie overeenkomt met het symbool `^` aan het begin van de regel, in plaats van aan het begin van de tekenreeks.

Het patroon `\*` komt overeen met een asteriskteken (de backslash wordt gebruikt om een letterlijke asterisk aan te geven in plaats van een kwantor `*`).

De ronde haakjes in de reguliere expressie definiëren een vastgelegde groep, en de methode `replace()` verwijst naar deze groep door gebruik te maken van de code `$1` in de vervangende tekenreeks. De markering `g` (`global`) in de reguliere expressie zorgt ervoor dat de methode `replace()` alle overeenkomsten in de tekenreeks (dus niet alleen de eerste) vervangt.

Wiki-patronen voor alinea's converteren

De methode `linesToParagraphs()` converteert elke regel in de ingevoerde Wiki-tekenreeks naar een HTML-tag `<p>` voor alinea's. Deze regels in de methode verwijderen lege regels van de ingevoerde Wiki-tekenreeks:

```
var pattern:RegExp = /^$/gm;
var result:String = input.replace(pattern, "");
```

De symbolen `^` en `$` van de reguliere expressie komen overeen met het begin en einde van een regel. De markering `m` (`multiline`) in de reguliere expressie zorgt ervoor dat de reguliere expressie overeenkomt met het `^`-symbool aan het begin van de regel, in plaats van het begin van de tekenreeks.

De methode `replace()` vervangt alle overeenkomende subtekenreeksen (lege regels) door een lege tekenreeks (`""`). De markering `g` (`global`) in de reguliere expressie zorgt ervoor dat de methode `replace()` alle overeenkomsten in de tekenreeks (dus niet alleen de eerste) vervangt.

URL's omzetten in HTML-tags <a >

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de gebruiker in de voorbeeldtoepassing op de knop Test klikt, nadat hij het selectievakje `urlToATag` heeft ingeschakeld, roept de toepassing de statische methode `URLParser.urlToATag()` aan om URL-tekenreeksen om te zetten van de ingevoerde Wiki-tekenreeks in HTML-tags `<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>");
```

De constructorfunctie `RegExp()` wordt gebruikt om een reguliere expressie (`urlPattern`) samen te stellen uit een aantal bestanddelen. Deze bestanddelen zijn elk een tekenreeks die een deel van het reguliere-expressiepatroon definiëren.

Het eerste deel van het reguliere-expressiepatroon, gedefinieerd door de tekenreeks `protocol`, definieert een URL-protocol: ofwel `http://` ofwel `ftp://`. De ronde haakjes definiëren een niet vastgelegde groep, die wordt aangegeven door het symbool `?`. Dit betekent dat de ronde haakjes eenvoudig worden gebruikt om een groep te definiëren voor het alternatievenpatroon `|`; de groep komt niet overeen met terugverwijzingscodes (`$1`, `$2`, `$3`) in de vervangende tekenreeks van de methode `replace()`.

De andere bestanddelen van de reguliere expressie maken elk gebruik van vastgelegde groepen (in het patroon aangegeven door ronde haakjes), die worden gebruikt in de terugverwijzingscodes (`$1`, `$2`, `$3`) in de vervangende tekenreeks van de methode `replace()`.

Het gedeelte van het patroon dat gedefinieerd is door de tekenreeks `urlPart` komt *ten minste* met een van de volgende tekens overeen: `a-z`, `0-9`, _ of `-`. De kwantor `+` geeft aan dat ten minste één teken overeenkomt. De `\.` geeft een vereist punt-teken (`aan.`) character. En de rest komt overeen met een andere tekenreeks van ten minste een van deze tekens: `a-z`, `0-9`, _ of `-`.

Het gedeelte van het patroon dat gedefinieerd is door de tekenreeks `optionalUrlPart` komt *nul keer of vaker* met een van de volgende tekens overeen: een punt (`.`) gevolgd door een willekeurig aantal alfanumerieke tekens (waaronder _ en `-`). De kwantor `*` geeft aan dat nul of meer tekens overeenkomen.

De aanroep van de methode `replace()` maakt gebruik van de reguliere expressie en stelt de vervangende HTML-tekenreeks samen, waarbij gebruik wordt gemaakt van terugverwijzingen.

De methode `urlToATag()` roept vervolgens de methode `emailToATag()` aan, die vergelijkbare technieken gebruikt om e-mailpatronen te vervangen door HTML-tekenreeksen `<a>` voor hyperlinks. De reguliere expressies die in dit voorbeeldbestand worden gebruikt om overeen te komen met HTTP-, FTP- en e-mail-URL's, zijn in deze voorbeelden vrij eenvoudig; er zijn veel gecompliceerdere reguliere expressies die ervoor zorgen dat dergelijke URL's correct overeenkomen.

Dollar-tekenreeksen omzetten in euro-tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de gebruiker in de testtoepassing op de knop Test klikt, nadat hij het selectievakje `dollarToEuro` heeft ingeschakeld, roept de toepassing de statische methode `CurrencyConverter.usdToEuro()` op om dollar-tekenreeksen (zoals `"$9.95"`) als volgt om te zetten in euro-tekenreeksen (zoals `"8.24 €"`):

```
var usdPrice:RegExp = /\$([\d,]+.\d+)+/g;
return input.replace(usdPrice, usdStrToEuroStr);
```

De eerste regel definieert een eenvoudig patroon voor overeenkomende dollar-tekenreeksen. Hierin wordt het teken `$` voorafgegaan door de backslash (`\`) als escape-teken.

De methode `replace()` gebruikt de reguliere expressie voor patroonovereenkomst, en roept de functie `usdStrToEuroStr()` aan om de vervangende tekenreeks (een waarde in euro) te bepalen.

Wanneer een functienaam gebruikt wordt als de tweede parameter van de methode `replace()`, worden de volgende delen als parameters doorgestuurd naar de aangeroepen functie:

- Het overeenkomende gedeelte van de tekenreeks.
- Elke willekeurige vastgelegde groep met ronde haakjes komt overeen. Het aantal op deze wijze doorgegeven argumenten hangt af van het aantal overeenkomsten van vastgelegde groepen tussen ronde haakjes. U kunt het aantal overeenkomsten van vastgelegde groepen tussen ronde haakjes vaststellen door `arguments.length - 3` binnen de functiecode te controleren.
- De indexpositie in de tekenreeks waar de overeenkomst begint.
- De volledige tekenreeks.

De methode `usdStrToEuroStr()` zet tekenreekspatronen in Amerikaanse dollar als volgt om in euro-tekenreeksen:

```
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;
}
```

De variabele `args[1]` vertegenwoordigt de vastgelegde groep tussen haakjes die overeenkomt met de reguliere expressie `usdPrice`. Dit is het numerieke gedeelte van de dollar-tekenreeks dat wil zeggen het dollarbedrag zonder het valutasymbool `$`. De methode past een wisselkoersconversie toe en retourneert de resulterende tekenreeks (met een navolgend €-symbool in plaats van een $-symbool ervóór).

Hoofdstuk 6: Werken met XML

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 bevat een groep klassen die is gebaseerd op de ECMAScript for XML (E4X) Specification (ECMA-357), 2nd Edition. Deze klassen bevatten krachtige en eenvoudig te gebruiken functionaliteit voor werken met XML-gegevens. Wanneer u E4X gebruikt, kunt u met XML-gegevens sneller codes ontwikkelen dan met de voorgaande programmeertechnieken. Een ander voordeel is dat de code die u produceert, eenvoudiger te lezen is.

Meer Help-onderwerpen

XML-klasse

ECMA-357-specificatie

Basisbeginselen van XML

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

XML is een standaardmanier om gestructureerde informatie weer te geven waar computers makkelijk mee kunnen werken en die voor personen eenvoudig te schrijven en te begrijpen is. XML is een afkorting voor eXtensible Markup Language. De XML-standaard is beschikbaar op www.w3.org/XML/.

XML biedt een standaard en gemakkelijke manier voor het indelen van gegevens, waardoor u deze eenvoudiger kunt lezen, benaderen en manipuleren. XML maakt gebruik van een boom- en tagstructuur die vergelijkbaar is met HTML. Hier volgt een eenvoudig voorbeeld van XML-gegevens:

```
<song>
    <title>What you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

XML-gegevens kunnen ook complexer zijn, met tags die zijn genest in andere tags maar ook in kenmerken en andere structurele elementen. Hier volgt een complexer voorbeeld van XML-gegevens:

```
<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>
```

Dit XML-document bevat tevens andere volledige XML-structuren (zoals de tags `song` met onderliggende items). Het document bevat bovendien andere XML-structuren zoals kenmerken (`tracknumber` en `length` in de tags `song`) en tags die andere tags bevatten in plaats van gegevens (zoals de tag `tracks`).

Aan de slag met XML

Voor personen die weinig of geen ervaring hebben met XML, volgt hierna een korte beschrijving van algemene aspecten van XML-gegevens. XML-gegevens worden geschreven als onbewerkte tekst met een specifieke syntaxis om de informatie te ordenen in een gestructureerde indeling. In het algemeen wordt één set van XML-gegevens een *XML-document* genoemd. In XML-indeling worden gegevens volgens een hiërarchische structuur geordend in *elementen* (dit kunnen enkele gegevensitems of containers voor andere elementen zijn). Elk XML-document heeft één element als hoofdniveau of hoofditem. Dit hoofdelement kan enkel informatie bevatten, maar het is gebruikelijker dat dit hoofdelement andere elementen bevat die vervolgens weer andere elementen bevatten, enzovoort. Het volgende XML-document bevat bijvoorbeeld de informatie over een muziekalbum:

```
<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>
```

Tussen elementen wordt onderscheid gemaakt door een set *tags*, waarbij de naam van het element is opgenomen tussen punthaken (kleiner dan- en groter dan-tekens). De openingstag geeft het begin van het element aan en heeft de naam van het element:

```
<title>
```

De afsluitingstag geeft het einde van het element aan en bevat een slash voor de naam van het element:

```
</title>
```

Wanneer een element geen inhoud bevat, kan het element worden geschreven als een leeg element (ook wel zelfsluitend element genoemd). In XML is dit element:

```
<lastplayed/>
```

gelijk aan dit element:

```
<lastplayed></lastplayed>
```

Naast de inhoud van het element tussen de openingstag en afsluitingstag, kan een element ook andere waarden, die ook wel *kenmerken* worden genoemd, bevatten die worden gedefinieerd in de openingstag van het element. In het volgende XML-element wordt bijvoorbeeld één kenmerk gedefinieerd met de naam `length` en met de waarde `"4:19"`:

```
<song length="4:19"></song>
```

Elk XML-element bevat inhoud, namelijk één waarde, een of meer XML-elementen of niets (in het geval van een leeg element).

Meer informatie over XML

Wanneer u meer informatie wilt over werken met XML, kunt u een aantal aanvullende boeken en bronnen over XML raadplegen, waaronder deze websites:

- XML-zelfstudie W3Schools: http://w3schools.com/xml/
- Zelfstudies voor XMLpitstop, discussielijsten en meer: http://xmlpitstop.com/

ActionScript-klassen voor het werken met XML

ActionScript 3.0 bevat diverse klassen die worden gebruikt voor het werken met XML-gestructureerde informatie. De twee hoofdklassen zijn als volgt:

- XML: vertegenwoordigt één XML-element dat een XML-document met meerdere onderliggende items of een element met één waarde kan zijn binnen een document.
- XMLList: vertegenwoordigt een set XML-elementen. Een object XMLList wordt gebruikt bij meerdere XML-elementen die 'items op hetzelfde niveau' (op hetzelfde niveau en met hetzelfde bovenliggende item in de hiërarchie van het XML-document) zijn. Een instantie XMLList is bijvoorbeeld de eenvoudigste manier om te werken met deze set XML-elementen (die waarschijnlijk is opgenomen in een XML-document):

  ```
  <artist type="composer">Fred Wilson</artist>
  <artist type="conductor">James Schmidt</artist>
  <artist type="soloist">Susan Harriet Thurndon</artist>
  ```

ActionScript bevat bovendien de klassen Namespace en QName voor geavanceerdere toepassingen met XML-naamruimten. Zie “XML-naamruimten gebruiken” op pagina 117 voor meer informatie.

Naast de ingebouwde klassen voor het werken met XML bevat ActionScript 3.0 ook diverse operatoren die specifieke functionaliteit bieden voor toegang tot XML-gegevens en voor het manipuleren van deze gegevens. Deze aanpak voor het werken met XML met deze klassen en operatoren wordt ECMAScript for XML (E4X) genoemd, zoals gedefinieerd in de ECMAScript for XML (E4X) Specification (ECMA-357), 2nd Edition.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen die u zult tegenkomen bij het programmeren van XML-afhandelingsroutines:

Element Eén item in een XML-document dat wordt geïdentificeerd als de inhoud tussen een starttag en een eindtag (inclusief de tags). XML-elementen kunnen tekstgegevens of andere elementen bevatten, of ze kunnen leeg zijn.

Leeg element Een XML-element die geen onderliggende elementen bevat. Lege elementen worden vaak geschreven als zelfsluitende tags (zoals `<element/>`).

Document Eén XML-structuur. Een XML-document kan een willekeurig aantal elementen bevatten (of bestaan uit slechts één leeg element). Een XML-document moet echter één element op hoofdniveau hebben dat alle andere elementen in het document bevat.

Knooppunt Een andere naam voor een XML-element.

Attribuut Een benoemde waarde met een element dat is geschreven in de openingstag van het element in de indeling `attributename='value'` en niet als een afzonderlijk onderliggend element dat in het element is genest.

De E4X-aanpak voor XML-verwerking

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De specificatie ECMAScript for XML definieert een set klassen en functionaliteit voor het werken met XML-gegevens. Deze klassen en functionaliteit worden samen *E4X genoemd.* ActionScript 3.0 bevat de volgende E4X-klassen: XML, XMLList, QName en Namespace.

De methoden, eigenschappen en operatoren van de E4X-klassen zijn ontworpen met de volgende doelstellingen:

- Eenvoud: waar mogelijk maakt E4X het schrijven en begrijpen van codes voor werken met XML-gegevens eenvoudiger.
- Consistentie: de methoden en de redenering achter E4X zijn intern consistent en zijn consistent met andere onderdelen van ActionScript.
- Vertrouwdheid: u kunt XML-gegevens manipuleren met bekende operatoren, zoals de puntoperator (.).

Opmerking: *ActionScript 2.0 heeft een andere XML-klasse. In ActionScript 3.0 is de naam van deze klasse gewijzigd in XMLDocument, zodat de naam niet conflicteert met de ActionScript 3.0 XML-klasse die deel uitmaakt van E4X.. In ActionScript 3.0 zijn de oudere klassen (XMLDocument, XMLNode, XMLParser en XMLTag) opgenomen in het pakket flash.xml met name ter ondersteuning van onderdelen uit oudere versies. De nieuwe E4X-klassen zijn kernklassen. U hoeft geen pakket te importeren om deze te kunnen gebruiken. Zie het flash.xml-pakket in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de oudere XML-klassen van ActionScript 2.0.*

Hierna volgt een voorbeeld van het manipuleren van gegevens met 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>
```

De toepassing laadt vaak XML-gegevens van een externe bron, zoals een webservice of een RSS-feed. Voor de duidelijkheid echter wijzen de codevoorbeelden die hier zijn opgegeven XML-gegevens toe als letterlijke gegevens.

Zoals wordt getoond in de volgende code, bevat E4X een aantal intuïtieve operatoren, zoals de puntoperator (.) en de operator voor kenmerkidentificatie (@), voor toegang tot eigenschappen en kenmerken in 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
```

U kunt de methode `appendChild()` gebruiken om een nieuw onderliggend knooppunt toe te wijzen aan XML, zoals in het volgende fragment wordt getoond:

```
var newItem:XML =
    <item id="3">
        <menuName>medium cola</menuName>
        <price>1.25</price>
    </item>

myXML.appendChild(newItem);
```

Gebruik de operatoren `@` en `.` niet alleen gebruiken om gegevens te lezen, maar ook om gegevens toe te wijzen, zoals in het volgende voorbeeld:

```
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";
```

U kunt als volgt een lus `for` gebruiken om knooppunten van XML te doorlopen:

```
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));
```

XML-objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een XML-object kan een XML-element, kenmerk, opmerking, verwerkingsinstructie of tekstelement vertegenwoordigen.

Een XML-object wordt geclassificeerd als een object met *eenvoudige inhoud* of als een object met *complexe inhoud*. Een XML-object dat onderliggende knooppunten bevat, wordt geclassificeerd als een object met complexe inhoud. Een XML-object bevat eenvoudige inhoud wanneer het een van de volgende elementen is: een kenmerk, een opmerking, een verwerkingsinstructie of een tekstknooppunt.

Het volgende XML-object bevat bijvoorbeeld complexe inhoud, inclusief een opmerking en een verwerkingsinstructie:

```
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>
```

Zoals in het volgende voorbeeld wordt getoond, kunt u nu de methoden `comments()` en `processingInstructions()` gebruiken om nieuwe XML-objecten, een opmerking en een verwerkingsinstructie te maken:

```
var x2:XML = x1.comments()[0];
var x3:XML = x1.processingInstructions()[0];
```

XML-eigenschappen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse XML heeft vijf statische eigenschappen:

- De eigenschappen `ignoreComments` en `ignoreProcessingInstructions` bepalen of opmerkingen of verwerkingsinstructies worden genegeerd wanneer het object XML wordt geparseerd.
- De eigenschap `ignoreWhitespace` bepaalt of witruimtetekens worden genegeerd in elementtags en ingesloten expressies die alleen door witruimtetekens worden gescheiden.
- De eigenschappen `prettyIndenten` `prettyPrinting` worden gebruikt om de tekst op te maken die wordt geretourneerd door de methoden `toString()` en `toXMLString()` van de klasse XML.

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over deze eigenschappen.

XML-methoden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de volgende methoden kunt u werken met de hiërarchische structuren van XML-objecten:

- `appendChild()`
- `child()`
- `childIndex()`
- `children()`
- `descendants()`
- `elements()`
- `insertChildAfter()`

- `insertChildBefore()`
- `parent()`
- `prependChild()`

Met de volgende methoden kunt u werken met kenmerken van XML-objecten:

- `attribute()`
- `attributes()`

Met de volgende methoden kunt u werken met eigenschappen van XML-objecten:

- `hasOwnProperty()`
- `propertyIsEnumerable()`
- `replace()`
- `setChildren()`

De volgende methoden zijn bedoeld voor het werken met gekwalificeerde namen en naamruimten:

- `addNamespace()`
- `inScopeNamespaces()`
- `localName()`
- `name()`
- `namespace()`
- `namespaceDeclarations()`
- `removeNamespace()`
- `setLocalName()`
- `setName()`
- `setNamespace()`

De volgende methoden zijn bedoeld voor het werken met en het vaststellen van bepaalde typen XML-inhoud:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `nodeKind()`
- `processingInstructions()`
- `text()`

De volgende methoden zijn bedoeld voor het omzetten in tekenreeksen en voor het opmaken van XML-objecten:

- `defaultSettings()`
- `setSettings()`
- `settings()`
- `normalize()`
- `toString()`
- `toXMLString()`

Er zijn een aantal aanvullende methoden:

- `contains()`
- `copy()`
- `valueOf()`
- `length()`

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over deze methoden.

XMLList-objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een instantie XMLList vertegenwoordigt een willekeurige verzameling van XML-objecten. Deze instantie kan volledige XML-documenten, XML-fragmenten of de resultaten van een XML-query bevatten.

Met de volgende methoden kunt u werken met de hiërarchische structuren van XMLList-objecten:

- `child()`
- `children()`
- `descendants()`
- `elements()`
- `parent()`

Met de volgende methoden kunt u werken met kenmerken van XMLList-objecten:

- `attribute()`
- `attributes()`

Met de volgende methoden kunt u werken met XMLList-eigenschappen:

- `hasOwnProperty()`
- `propertyIsEnumerable()`

De volgende methoden zijn bedoeld voor het werken met en het vaststellen van bepaalde typen XML-inhoud:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `processingInstructions()`
- `text()`

De volgende methoden zijn bedoeld voor het omzetten in tekenreeksen en voor het opmaken van het XMLList-object:

- `normalize()`
- `toString()`
- `toXMLString()`

Er zijn een aantal aanvullende methoden:

- `contains()`

- `copy()`
- `length()`
- `valueOf()`

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over deze methoden.

U kunt voor een object XMLList dat precies één XML-element bevat, alle eigenschappen en methoden van de klasse XML gebruiken, omdat een XMLList met een XML-element op dezelfde wijze wordt behandeld als een object XML. U kunt bijvoorbeeld in de volgende code de methode `appendChild()` van de klasse XML gebruiken, omdat `doc.div` een object XMLList is dat één element bevat:

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

Zie "XML-objecten" op pagina 106 voor een lijst met XML-eigenschappen en -methoden.

XML-variabelen initialiseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt als volgt een letterlijke XML toewijzen aan een object XML:

```
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>
```

Zoals in het volgende fragment wordt getoond, kunt u ook de constructor `new` gebruiken om een instantie van een object XML te maken van een tekenreeks die XML-gegevens bevat:

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                            + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

Wanneer de XML-gegevens in de tekenreeks niet correct zijn samengesteld (wanneer bijvoorbeeld een afsluitingstag ontbreekt), wordt een uitvoeringsfout weergegeven.

U kunt gegevens ook via verwijzing (van andere variabelen) in een object XML doorgeven, zoals in het volgende voorbeeld wordt getoond:

```
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>
```

Met de klasse URLLoad kunt u XML-gegevens van een URL laden, zoals in het volgende voorbeeld wordt getoond:

```
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!");
    }
}
```

Met de klasse XMLSocket kunt u XML-gegevens lezen van een socketverbinding. Zie de beschrijving van de klasse XMLSocket in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

XML-objecten samenstellen en transformeren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `prependChild()` of de methode `appendChild()` kunt u een eigenschap toevoegen aan het begin of einde van een lijst met eigenschappen van een object XML, zoals in het volgende voorbeeld wordt getoond:

```
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>
```

U kunt als volgt de methode `insertChildBefore()` of de methode `insertChildAfter()` gebruiken om een eigenschap toe te voegen voor of na een bepaalde eigenschap:

```
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>)
```

Zoals in het volgende voorbeeld wordt getoond, kunt u ook de accoladeoperatoren ({ en }) gebruiken om gegevens via verwijzing (van andere variabelen) door te geven bij het samenstellen van XML-objecten:

```
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)
}
```

U kunt eigenschappen en kenmerken toewijzen aan een object XML met de operator =, zoals in het volgende voorbeeld:

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

Hiermee wordt het XML-object `x` ingesteld op het volgende:

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

Met de operatoren + en += kunt u XMLList-objecten samenvoegen.

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

Hiermee wordt het XMLList-object `xList` ingesteld op het volgende:

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

XML-structuren doorlopen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een van de krachtige kenmerken van XML is de mogelijkheid om complexe, geneste gegevens te bieden via een lineaire tekenreeks van teksttekens. Wanneer u gegevens laadt in een object XML, worden de gegevens door ActionScript geparseerd en geladen in de hiërarchische structuur in het geheugen (of er wordt een uitvoeringsfout verzonden wanneer de XML-gegevens niet correct zijn samengesteld).

Met de operatoren en methoden van de XML- en XMLList-objecten kunt u eenvoudig de structuur van XML-gegevens doorlopen.

U kunt de puntoperator (.) en de operator .. (afstammende accessor) gebruiken om toegang te krijgen tot onderliggende eigenschappen van een object XML. Bekijk het volgende XML-object:

```
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>
```

Het object `myXML.book` is een XMLList-object dat onderliggende eigenschappen bevat van het object `myXML` met de naam `book`. Dit zijn twee XML-objecten die overeenkomen met de twee eigenschappen `book` van het object `myXML`.

Het object `myXML..lastName` is een XMLList-object dat een willekeurige afstammende eigenschap bevat met de naam `lastName`. Dit zijn twee XML-objecten die overeenkomen met de twee eigenschappen `lastName` van het object `myXML`.

Het object `myXML.book.editor.lastName` is een XMLList-object dat een willekeurig onderliggend item bevat met de naam `lastName` van onderliggende items met de naam `editor` van onderliggende items met de naam `book` van het object `myXML`: in dit geval een XMLList-object dat slechts een object XML bevat (de eigenschap `lastName` met de waarde `Case`).

Toegang verkrijgen tot bovenliggende en onderliggende knooppunten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `parent()` retourneert het bovenliggende item van een object XML.

U kunt de ordinale indexwaarden van een lijst met onderliggende items gebruiken om toegang te krijgen tot bepaalde onderliggende objecten. Bekijk bijvoorbeeld een XML-object `myXML` dat twee onderliggende eigenschappen heeft met de naam `book`. Elke onderliggende eigenschap met de naam `book` heeft een indexnummer dat hieraan is gekoppeld:

```
myXML.book[0]
myXML.book[1]
```

U kunt indexnummers opgeven voor de naam van het onderliggende item van zowel het eerste als het tweede niveau om toegang te krijgen tot een bepaald onderliggend item op het tweede niveau:

```
myXML.book[0].title[0]
```

Wanneer er echter maar één onderliggend item is van `x.book[0]` met de naam `title`, kunt u de verwijzing naar de index als volgt weglaten:

```
myXML.book[0].title
```

Op dezelfde manier kunt u als volgt beide verwijzingen naar de index weglaten wanneer er slechts één onderliggend item book is van het object `x` en wanneer dat onderliggende object slechts één object title heeft:

```
myXML.book.title
```

U kunt de methode `child()` gebruiken om naar onderliggende items te navigeren met namen die zijn gebaseerd op een variabele of een expressie, zoals in het volgende voorbeeld wordt getoond:

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

Toegang verkrijgen tot kenmerken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met het symbool `@` (de operator voor kenmerkidentificatie) krijgt u toegang tot kenmerken in een object XML of een object XMLList, zoals wordt getoond in de volgende code:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

U kunt het jokersymbool `*` gebruiken samen met het symbool `@` om toegang te krijgen tot alle kenmerken van een object XML of een object XMLList, zoals wordt getoond in de volgende code:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

Met de methoden `attribute()` of `attributes()` krijgt u toegang tot een specifiek kenmerk of tot alle kenmerken van een object XML of een object XMLList, zoals in de volgende code:

```
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
```

Met de volgende syntaxis krijgt u bovendien toegang tot kenmerken, zoals in het volgende voorbeeld wordt getoond:

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

Elk van deze is gelijk aan `employee.@id`. De syntaxis `employee.@id` heeft echter de voorkeur.

Filteren op kenmerk- of elementwaarde

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de ronde-haakjeoperatoren, `(` en `)`) gebruiken om elementen met een specifieke elementnaam of kenmerkwaarde te filteren. Bekijk het volgende XML-object:

```
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>
```

De volgende expressies zijn allemaal geldig:

- `x.employee.(lastName == "McGee")`: dit is het tweede knooppunt `employee`.
- `x.employee.(lastName == "McGee").firstName`: dit is de eigenschap `firstName` van het tweede knooppunt `employee`.
- `x.employee.(lastName == "McGee").@id`: dit is de waarde van het kenmerk `id` van het tweede knooppunt `employee`.
- `x.employee.(@id == 347)`: het eerste knooppunt `employee`.
- `x.employee.(@id == 347).lastName`: dit is de eigenschap `lastName` van het eerste knooppunt `employee`.
- `x.employee.(@id > 300)`: dit is een XMLList met beide eigenschappen `employee`.

- `x.employee.(position.toString().search("analyst") > -1)`: dit is een XMLList met beide eigenschappen `position`.

Als u probeert te filteren op kenmerken of elementen die niet bestaan, treedt een uitzondering op. De laatste regel van de volgende code genereert bijvoorbeeld een fout omdat er geen kenmerk `id` bestaat in het tweede element `p`:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

Op dezelfde manier genereert de laatste regel van de volgende code een fout omdat er geen eigenschap `b` bestaat van het tweede element `p`:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

U kunt deze fouten vermijden door de eigenschappen die overeenkomende kenmerken of elementen hebben, te identificeren met de methoden `attribute()` en `elements()`, zoals in de volgende code:

```
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'));
```

U kunt ook de methode `hasOwnProperty()` gebruiken, zoals in de volgende code:

```
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'));
```

De instructies for..in en for each..in gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 biedt ondersteuning voor de instructies `for..in` en `for each..in` om XMLList-objecten te doorlopen. Bekijk bijvoorbeeld het volgende XML-object (`myXML`) en het XMLList-object (`myXML.item`). Het XMLList-object (`myXML.item`) bestaat uit twee knooppunten `item` van het XML-object.

```
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>;
```

Met de lus `for..in` kunt u een set eigenschapnamen doorlopen in een XMLList:

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

Met de lus `for each..in` kunt u de eigenschappen doorlopen in de XMLList:

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

XML-naamruimten gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naamruimten in een XML-object (of XML-document) identificeren het type gegevens dat het object bevat. Bij het verzenden en leveren van XML-gegevens naar een webservice die gebruikmaakt van het SOAP-berichtenprotocol, declareert u bijvoorbeeld de naamruimte in de openingstag van 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>;
```

De naamruimte heeft een voorvoegsel (`soap`) en een URI die de naamruimte definieert `http://schemas.xmlsoap.org/soap/envelope/`).

ActionScript 3.0 bevat de klasse Namespace voor werken met XML-naamruimten. U kunt de klasse Namespace voor het object XML in het vorige voorbeeld als volgt gebruiken:

```
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";
```

De klasse XML bevat de volgende methoden voor het werken met naamruimten: `addNamespace()`, `inScopeNamespaces()`, `localName()`, `name()`, `namespace()`, `namespaceDeclarations()`, `removeNamespace()`, `setLocalName()`, `setName()` en `setNamespace()`.

Met de compileraanwijzing `default xml namespace` kunt u een standaardnaamruimte toewijzen voor XML-objecten. In het volgende voorbeeld hebben zowel `x1` als `x2` dezelfde standaardnaamruimte:

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

XML-typeomzetting

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt XML-objecten en XMLList-objecten omzetten in tekenreekswaarden. Op dezelfde manier kunt u tekenreeksen omzetten in XML-objecten en XMLList-objecten. Houd er rekening mee dat alle XML-kenmerkwaarden, -namen en -tekstwaarden ook tekenreeksen zijn. In de volgende secties worden al deze vormen van XML-typeomzetting beschreven.

Objecten XML en XMLList omzetten in tekenreeksen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klassen XML en XMLList bevatten een methode `toString()` en een methode `toXMLString()`. De methode `toXMLString()` retourneert een tekenreeks die alle tags, kenmerken, naamruimtedeclaraties en inhoud van het object XML bevat. De methode `toString()` van XML-objecten met complexe inhoud (onderliggende elementen) heeft precies dezelfde werking als de methode `toXMLString()`. De methode `toString()` van XML-objecten met eenvoudige inhoud (objecten met slechts één tekstelement) retourneert alleen de tekstinhoud van het element, zoals in het volgende voorbeeld wordt getoond:

```
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
```

Wanneer u de methode `trace()` gebruikt zonder `toString()` of `toXMLString()` op te geven, worden de gegevens standaard omgezet met de methode `toString()`, zoals wordt getoond in deze code:

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

Wanneer u de methode `trace()` gebruikt om fouten op te sporen in codes, is het nuttig om de methode `toXMLString()` te gebruiken voor een optimalere uitvoer van de methode `trace()`.

Tekenreeksen omzetten in XML-objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt als volgt de constructor `new XML()` gebruiken om een object XML te maken van een tekenreeks:

```
var x:XML = new XML("<a>test</a>");
```

Wanneer u een tekenreeks probeert om te zetten in XML van een tekenreeks die ongeldige XML vertegenwoordigt of die XML vertegenwoordigt die niet correct is samengesteld, wordt als volgt een uitvoeringsfout gegenereerd:

```
var x:XML = new XML("<a>test"); // throws an error
```

Kenmerkwaarden, -namen en -tekstwaarden van tekenreeksen omzetten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Alle XML-kenmerkwaarden, -namen en -tekstwaarden zijn tekenreeksgegevenstypen die u mogelijk moet omzetten in andere gegevenstypen. In de volgende code wordt bijvoorbeeld de functie `Number()` gebruikt om tekstwaarden om te zetten in getallen:

```
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;
```

Wanneer in deze code geen gebruik zou zijn gemaakt van de functie `Number()` zou de operator + worden geïnterpreteerd als tekenreekssamenvoegingsoperator en zou de methode `trace()` in de laatste regel het volgende genereren:

```
01.003.95
```

Externe XML-documenten lezen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse URLLoader gebruiken om XML-gegevens te laden van een URL. U kunt de volgende code in de toepassingen gebruiken door de waarde `XML_URL` in het voorbeeld te vervangen door een geldige URL:

```
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.");
}
```

U kunt bovendien de klasse XMLSocket gebruiken om een asynchrone XML-socketverbinding te maken met een server. Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Voorbeeld van XML in ActionScript: RSS-gegevens van internet laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de voorbeeldtoepassing van RSSViewer worden een aantal kenmerken van werken met XML in ActionScript getoond, inclusief de volgende kenmerken:

- XML-methoden gebruiken voor het doorlopen van XML-gegevens in de vorm van een RSS-feed.
- XML-methoden gebruiken voor het samenstellen van XML-gegevens in de vorm van HTML voor gebruik in een tekstveld.

De RSS-notatie wordt veelal gebruikt voor het publiceren van nieuws via XML. Een eenvoudig RSS-gegevensbestand kan er als volgt uitzien:

```
<?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>
```

De toepassing SimpleRSS leest RSS-gegevens van internet, parseert de gegevens voor koppen (titels), koppelingen en beschrijvingen en retourneert deze gegevens. De klasse SimpleRSSUI biedt de UI en roept de klasse SimpleRSS aan die volledige XML-verwerking uitvoert.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van RSSViewer vindt u in de map Samples/RSSViewer. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
RSSViewer.mxml of RSSViewer.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/rssViewer/RSSParser.as	Een klasse met methoden die gebruikmaken van E4X voor het doorlopen van RSS-gegevens (XML) en het genereren van een overeenkomende HTML-representatie.
RSSData/ak.rss	Een RSS-voorbeeldbestand. De toepassing is ingesteld om RSS-gegevens te lezen van het web van een RSS-feed van Flex die door Adobe wordt aangeboden. U kunt de toepassing echter gemakkelijk wijzigen voor het lezen van RSS-gegevens uit dit document. Hierbij wordt een iets ander schema gebruikt dan bij de RSS-feed van Flex.

XML-gegevens lezen en parseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse RSSParser bevat een methode `xmlLoaded()` waarmee de RSS-invoergegevens die zijn opgeslagen in de variabele `rssXML`, worden omgezet in een tekenreeks met uitvoer die is opgemaakt in HTML, `rssOutput`.

Aan het begin van de methode wordt de standaard XML-naamruimte via de code ingesteld wanneer de RSS-brongegevens een standaardnaamruimte bevatten:

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

De volgende regels doorlopen de inhoud van de XML-brongegevens en onderzoeken elke afstammende eigenschap met de naam `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);
}
```

Met de eerste drie regels worden tekenreeksvariabelen ingesteld voor de weergave van de titel, beschrijving en koppelingseigenschappen van de eigenschap `item` van de XML-gegevens. Met de volgende regel wordt de methode `buildItemHTML()` aangeroepen voor het omzetten van HTML-gegevens in een object XMLList met behulp van de drie nieuwe tekenreeksvariabelen als parameters.

XMLList-gegevens samenstellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De HTML-gegevens (een object XMLList) hebben de volgende notatie:

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

Met de eerste regels van de methode wordt de standaard XML-naamruimte verwijderd:

```
default xml namespace = new Namespace();
```

De compileraanwijzing `default xml namespace` heeft een bereik op het niveau van functieblokken. Dit houdt in dat het bereik van deze declaratie de methode `buildItemHTML()` is.

Met de regels die hierna volgen, wordt het XMLList samengesteld op grond van de tekenreeksargumenten die zijn doorgegeven aan de functie:

```
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;
```

Dit object XMLList vertegenwoordigt tekenreeksgegevens die geschikt zijn voor een HTML-tekstveld van ActionScript.

De methode `xmlLoaded()` maakt gebruik van de geretourneerde waarde van de methode `buildItemHTML()` en zet deze om in een tekenreeks:

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

De titel van de RSS-feed extraheren en een aangepaste gebeurtenis verzenden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `xmlLoaded()` wordt een tekenreeksvariabele `rssTitle` ingesteld op grond van informatie in de RSS XML-brongegevens:

```
rssTitle = rssXML.channel.title.toString();
```

Tot slot genereert de methode `xmlLoaded()` een gebeurtenis die aan de toepassing meldt dat de gegevens zijn geparseerd en dat deze beschikbaar zijn:

```
            dataWritten = new Event("dataWritten", true);
```

Hoofdstuk 7: Native JSON-functionaliteit gebruiken

ActionScript 3.0 beschikt over een native API voor het coderen en decoderen van ActionScript-objecten met gebruik van de JSON-indeling (JavaScript Object Notation). De JSON-klasse en ondersteunende lidfuncties volgen de vijfde uitgave van de ECMA-262-specificatie, afgezien van enkele afwijkingen.

Todd Anderson, een lid van de Community, verschaft een vergelijking tussen de native JSON-API en de door een derde partij ontwikkelde JSON-klasse as3corelib. Zie Met native JSON werken in Flash Player 11.

Meer Help-onderwerpen

JSON

Overzicht van de JSON API

De ActionScript JSON-API bestaat uit de JSON-klasse en de `toJSON()`-lidfuncties voor enkele native klassen. Voor toepassingen die voor bepaalde klassen een aangepaste JSON-codering vereisen, verschaft het ActionScript-framework manieren om de standaardcodering te overschrijven.

De JSON-klasse verwerkt intern het importeren en exporteren van alle ActionScript-klassen die geen `toJSON()` -lid verschaffen. In dergelijke gevallen doorloopt JSON de openbare eigenschappen van elk object dat het tegenkomt. Als een object andere objecten bevat, keert JSON terug naar de geneste objecten en wordt hetzelfde traject doorlopen. Als een object de methode `toJSON()` biedt, gebruikt JSON die aangepaste methode in plaats van het interne algoritme.

De JSON-interface bestaat uit de coderingsmethode `stringify()` en de decoderingsmethode `parse()`. Elke methode verschaft een parameter waarmee u uw eigen logica kunt invoegen in de JSON-workflow voor coderen of decoderen. Voor `stringify()` heet deze parameter `replacer` en voor `parse()` heet de parameter `reviver`. Deze parameters hebben een functiedefinitie met twee argumenten die de volgende handtekening gebruiken:

```
function(k, v):*
```

toJSON()-methoden

De handtekening voor `toJSON()-methoden` is

```
public function toJSON(k:String):*
```

`JSON.stringify()` roept `toJSON()` aan, indien deze bestaat, voor elke publieke eigenschap die deze tegenkomt tijdens het doorlopen van een object. Een eigenschap bestaat uit een sleutelwaardepaar. Wanneer `stringify()` `toJSON()` aanroept, geeft het de sleutel `k` door van de eigenschap die op dat moment wordt onderzocht. Een typische `toJSON()`-implementatie evalueert elke eigenschapsnaam en retourneert de gewenste codering van de waarde van deze eigenschap.

De methode `toJSON()` kan een waarde van elk type (aangeduid als *) retourneren, niet alleen een tekenreeks. Dankzij dit variabele type retournering kan `toJSON()`, indien van toepassing, een object retourneren. Als een eigenschap van uw aangepaste klasse bijvoorbeeld een object uit een andere externe bibliotheek bevat, kunt u dat object retourneren als `toJSON()` uw eigenschap aantreft. JSON keert dan terug naar het externe object. De flow van het coderingsproces gaat als volgt:

- Als `toJSON()` een object retourneert dat niet wordt geëvalueerd als een tekenreeks, keert `stringify()` terug naar dat object.
- Als `toJSON()` een tekenreeks retourneert, wordt die waarde door `stringify()` opgenomen in een andere tekenreeks, die vervolgens wordt geretourneerd. Hierna wordt doorgegaan naar de volgende waarde.

In veel gevallen verdient het de voorkeur dat een object wordt geretourneerd in plaats van een JSON-tekenreeks die door uw toepassing is gemaakt. Bij het retourneren van een object wordt het geïntegreerde JSON-coderingsalgoritme toegepast en kan JSON ook geneste objecten recursief doorlopen.

De `toJSON()`-methode is niet gedefinieerd in de Object-klasse, noch in de meeste andere native klassen. Als deze methode ontbreekt, weet JSON dat het standaard doorlopen van de publieke eigenschappen van het object moet worden uitgevoerd. Indien gewenst, kunt u `toJSON()` ook gebruiken om de privé-eigenschappen van uw object te onthullen.

Enkele native klassen veroorzaken problemen die de ActionScript-bibliotheken niet in alle gevallen op effectieve wijze kunnen oplossen. Voor deze klassen biedt ActionScript een triviale implementatie die clients al naar gelang hun behoeften opnieuw kunnen implementeren. De volgende klassen verschaffen eenvoudige `toJSON()`-leden:

- ByteArray
- Date
- Dictionary
- XML

U kunt een subklasse maken van de klasse ByteArray om de methode `toJSON()` van deze klasse te overschrijven, of u kunt het prototype van de klasse opnieuw definiëren. De Date- en XML-klassen die als definitief worden gedeclareerd, vereisen dat u het prototype van de klasse gebruikt om `toJSON()` opnieuw te definiëren. De klasse Dictionary wordt als dynamisch gedeclareerd, zodat u meer vrijheid hebt om `toJSON()` te overschrijven.

Aangepast JSON-gedrag definiëren

Als u uw eigen JSON-codering en decodering wilt implementeren voor native klassen, hebt u de volgende mogelijkheden:

- `toJSON()` definiëren of overschrijven voor uw aangepaste subklasse van een niet-definitieve native klasse
- `toJSON()` definiëren of opnieuw definiëren voor het klassenprototype
- De eigenschap `toJSON` definiëren voor een dynamische klasse
- De parameters `JSON.stringify() replacer` en `JSON.parser() reviver` gebruiken

Meer Help-onderwerpen

ECMA-262, vijfde editie

toJSON() definiëren voor het prototype van een geïntegreerde klasse

De native JSON-implementatie in ActionScript spiegelt het ECMAScript JSON-mechanisme dat is gedefinieerd in de vijfde editie van ECMA-262. Aangezien ECMAScript geen ondersteuning biedt voor klassen, definieert ActionScript JSON-gedrag in termen van op prototype gebaseerde verzending. Prototypen zijn voorlopers van ActionScript 3.0-klassen waarmee gesimuleerde overerving, het toevoegen van leden en herdefinities mogelijk worden.

U kunt in ActionScript `toJSON()` definiëren of opnieuw definiëren als het prototype van een willekeurige klasse. Dit recht is zelfs van toepassing op klassen die als definitief zijn gemarkeerd. Wanneer u `toJSON()` definieert voor een klasseprototype, geldt uw definitie voor alle instanties van die klasse binnen het bereik van uw toepassing. Hier ziet u bijvoorbeeld hoe u een `toJSON()`-methode kunt definiëren voor het prototype MovieClip:

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

Als uw toepassing vervolgens `stringify()` aanroept voor een willekeurige MovieClip-instantie, retourneert `stringify()` de uitvoer van uw`toJSON()` -methode:

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

Het is ook mogelijk `toJSON()` te overschrijven in de native klassen die de methode definiëren. Met de volgende code overschrijft u bijvoorbeeld `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"
```

toJSON() definiëren of overschrijven op klassenniveau

Toepassingen zijn niet altijd verplicht prototypen te gebruiken om `toJSON()` opnieuw te definiëren. U kunt `toJSON()` ook definiëren als lid van een subklasse als de bovenliggende klasse niet is gemarkeerd als definitief. U kunt bijvoorbeeld de klasse ByteArray uitbreiden en een openbare `toJSON()`-functie definiëren:

```
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"
```

In geval van dynamische klassen kunt u een `toJSON`-eigenschap toevoegen aan een object van die klasse en er als volgt een functie aan toewijzen:

```
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."}
```

U kunt `toJSON()` voor elke ActionScript-klasse overschrijven, definiëren of opnieuw definiëren. De meeste geïntegreerde ActionScript-klassen definiëren `toJSON()` echter niet. De Object-klasse definieert `toJSON` niet in het standaardprototype en declareert het niet als een klassenlid. Slechts enkele native klassen definiëren de methode als een prototypefunctie. In de meeste klassen is het dus niet mogelijk om `toJSON()` in de traditionele betekenis van het woord te overschrijven.

Native klassen die geen definitie bieden voor `toJSON()` worden geserialiseerd met betrekking tot JSON door de interne JSON-implementatie. Vervang deze geïntegreerde functionaliteit waar mogelijk niet. Als u een lid van `toJSON()` definieert, wordt uw logica gebruikt door de JSON-klasse, en niet de eigen functionaliteit van deze klasse.

De replacer-parameter JSON.stringify() gebruiken

Het is handig `toJSON()` te overschrijven voor het prototype als u het JSON-exportgedrag van een klasse in de gehele toepassing wilt wijzigen. In bepaalde gevallen is het echter mogelijk dat uw exportlogica alleen van toepassing is op speciale gevallen onder overgangsvoorwaarden. Voor dergelijke kleine wijzigingen kunt u de parameter `replacer` van de methode `JSON.stringify()` gebruiken.

De methode `stringify()` past de functie die is doorgegeven via de parameter `replacer` toe op het object dat wordt gecodeerd. De handtekening voor deze functie is vergelijkbaar met die van `toJSON()`:

```
function (k,v):*
```

In tegenstelling tot `toJSON()` vereist de functie `replacer` de waarde `v` plus de sleutel `k`. Dit verschil is nodig omdat `stringify()` wordt gedefinieerd voor het statische JSON-object in plaats van voor het object dat wordt gecodeerd. Als `JSON.stringify()` `replacer(k,v)` aanroept, doorloopt het het oorspronkelijke invoerobject. De impliciete parameter `this` die aan de functie `replacer` is doorgegeven, verwijst naar het object dat de sleutel en waarde bevat. Aangezien `JSON.stringify()` het oorspronkelijke invoerobject niet wijzigt, blijft dat object ongewijzigd in de container die wordt doorlopen. U kunt daarom de code `this[k]` gebruiken om een query uit te voeren naar de sleutel voor het oorspronkelijke object. De parameter `v` bevat de waarde die `toJSON()` omzet.

Net als `toJSON()` kan de functie `replacer` elk mogelijke type waarde retourneren. Als een `replacer` een tekenreeks retourneert, kan de JSON-engine de inhoud tussen escape-tekens plaatsen en deze inhoud tussen escape-tekens vervolgens tussen aanhalingstekens plaatsen. Deze manier van insluiten garandeert dat `stringify()` een geldig JSON-tekenreeksobject ontvangt dat een tekenreeks blijft in de volgende aanroep naar `JSON.parse()`.

De volgende code gebruikt de parameter `replacer` en de impliciete parameter `this` om de waarden `time` en `hours` van een Date-object te retourneren:

```
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;
});
```

De JSON.parse()-parameter reviver gebruiken

De parameter `reviver` van de methode `JSON.parse()` doet het tegenovergestelde van de functie `replacer`: de parameter zet een JSON-tekenreeks om in een bruikbaar ActionScript-object. Het argument `reviver` is een functie die twee parameters gebruikt en een willekeurig type retourneert:

```
function (k,v):*
```

In deze functie is `k` een sleutel en is `v` de waarde van `k`. Net als `stringify()` doorloopt `parse()` de JSON-sleutelwaardeparen en past deze de functie `reviver` (als deze bestaat) toe op elk paar. Een mogelijk probleem is dat de JSON-klasse de ActionScript-klassenaam van een object niet uitvoert. Het kan dus lastig zijn te weten welk type object moet worden vernieuwd. Dit kan vooral lastig zijn in geval van geneste objecten. Tijdens het ontwerpen van de functies `toJSON()`, `replacer` en `reviver` kunt u manieren bedenken voor het identificeren van de ActionScript-objecten die worden geëxporteerd terwijl de originele objecten intact blijven.

Voorbeeld van parseren

Het volgende voorbeeld illustreert een strategie voor het vernieuwen van objecten die zijn geparseerd van JSON-tekenreeksen. In dit voorbeeld worden de volgende twee klassen gedefinieerd: JSONGenericDictExample en JSONDictionaryExtnExample. JSONGenericDictExample is een aangepaste Dictionary-klasse. Elke record bevat de naam en geboortedatum van een persoon, plus een unieke id. Steeds wanneer de constructor JSONGenericDictExample wordt aangeroepen, wordt het net gemaakte object toegevoegd aan een interne statische array met als id een statisch toenemend geheel getal. De klasse JSONGenericDictExample definieert ook de een methode `revive()` die alleen het gedeelte met het gehele getal extraheert uit het langere `id`-lid. De methode `revive()` gebruikt dit gehele getal om het juiste vernieuwbare object op te zoeken en te retourneren.

De klasse JSONDictionaryExtnExample vormt een uitbreiding van de klasse Dictionary van ActionScript. De records van deze klasse hebben geen ingestelde structuur en kunnen willekeurige gegevens bevatten. Gegevens worden toegewezen nadat een JSONDictionaryExtnExample-object is samengesteld in plaats van als eigenschappen die door de klasse zijn gedefinieerd. JSONDictionaryExtnExample legt de gebruikte JSONGenericDictExample-objecten vast als sleutels. Wanneer een JSONDictionaryExtnExample-object wordt vernieuwd, gebruikt de functie `JSONGenericDictExample.revive()` de id die aan JSONDictionaryExtnExample is gekoppeld om het juiste sleutelobject op te halen.

Het belangrijkste is echter dat de methode `JSONDictionaryExtnExample.toJSON()` in aanvulling op het JSONDictionaryExtnExample-object ook een markeringsreeks retourneert. Deze reeks identificeert de JSON-uitvoer als onderdeel van de klasse JSONDictionaryExtnExample. Deze markering neemt alle twijfel weg over het type object dat tijdens `JSON.parse()` wordt verwerkt.

```
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;
        }
    }
}
```

Wanneer de volgende runtimescript `JSON.parse()` aanroept voor een JSONDictionaryExtnExample-object, roept de `reviver`-functie `JSONGenericDictExample.revive()` aan voor elk object in JSONDictionaryExtnExample. Bij deze aanroep wordt de id die staat voor de objectsleutel geëxtraheerd. Deze id wordt door de functie `JSONGenericDictExample.revive()` gebruikt om het opgeslagen JSONDictionaryExtnExample-object op te halen van een statische privé-array en het vervolgens te retourneren.

```
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]>]
```

Hoofdstuk 8: Gebeurtenissen afhandelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met een systeem voor gebeurtenisafhandeling kunnen programmeurs op een gemakkelijke manier reageren op gebruikersinvoer en systeemgebeurtenissen. Het gebeurtenismodel van ActionScript 3.0 is niet alleen handig, maar voldoet ook aan de standaarden en is geïntegreerd in de weergavelijst. Het nieuwe gebeurtenismodel is gebaseerd op de Document Object Model (DOM) Level 3 Events Specification, een industriestandaard voor gebeurtenisafhandelingsarchitectuur, en bevat een krachtig en toch gebruiksvriendelijk gebeurtenisafhandelingsmechanisme voor ActionScript-programmeurs.

Het ActionScript 3.0-systeem voor gebeurtenisafhandeling werkt nauw samen met de weergavelijst. Lees om de basisbeginselen van de weergavelijst te begrijpen "Weergave programmeren" op pagina 161.

Meer Help-onderwerpen

flash.events-pakket

DOM (Document Object Model) Level 3 Events Specification

Basisbeginselen van gebeurtenisafhandeling

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een gebeurtenis zien als elk soort gebeurtenis in een SWF-bestand die voor u als programmeur van belang kan zijn. Zo wordt door de meeste SWF-bestanden een bepaald soort gebruikersinteractie ondersteund. Dit kan iets eenvoudigs zijn, zoals het reageren op een muisklik, of iets complexers, zoals het accepteren en verwerken van gegevens die in een formulier zijn ingevuld. Dergelijke gebruikersinteractie met het SWF-bestand wordt altijd als een gebeurtenis beschouwd. Gebeurtenissen kunnen ook zonder directe gebruikersinteractie plaatsvinden, zoals wanneer het laden van gegevens van een server is voltooid of wanneer een camera die op het systeem is aangesloten, actief is geworden.

In ActionScript 3.0 wordt elke gebeurtenis door een gebeurtenisobject vertegenwoordigd, wat een instantie is van de klasse Event of van een van de subklassen van deze klasse. Een gebeurtenisobject bevat niet alleen informatie over een specifieke gebeurtenis, maar ook methoden die de bewerking van het gebeurtenisobject vergemakkelijken. Als Flash Player of AIR bijvoorbeeld een muisklik detecteert, wordt een gebeurtenisobject gemaakt (een instantie van de klasse MouseEvent), die deze specifieke muisklikgebeurtenis vertegenwoordigt.

Nadat het gebeurtenisobject is gemaakt, wordt dit door Flash Player of AIR *verzonden*. Dit houdt in dat het gebeurtenisobject wordt doorgegeven aan het object dat het doel is van de gebeurtenis. Een object dat als doel voor een verzonden gebeurtenis fungeert, wordt een *gebeurtenisdoel* genoemd. Als bijvoorbeeld een camera die op het systeem is aangesloten, actief is geworden, wordt door Flash Player een gebeurtenisobject direct naar het gebeurtenisdoel verzonden, wat in dit geval het object is dat de camera vertegenwoordigt. Als het gebeurtenisdoel zich echter in het weergaveoverzicht bevindt, loopt het gebeurtenisobject van boven naar beneden door de hiërarchie van het weergaveoverzicht, totdat de doelgebeurtenis is bereikt. In sommige gevallen loopt het gebeurtenisobject vervolgens weer via dezelfde route omhoog door de hiërarchie van het weergaveoverzicht. Dit doorlopen van de hiërarchie van het weergaveoverzicht wordt de *gebeurtenisstroom* genoemd.

U kunt met behulp van gebeurtenislisteners naar gebeurtenisobjecten in de code luisteren. *Gebeurtenislisteners zijn de functies of methoden die u schrijft om op specifieke gebeurtenissen te reageren.* Als u er zeker van wilt zijn dat het programma op gebeurtenissen reageert, moet u gebeurtenislisteners toevoegen aan het gebeurtenisdoel of aan een willekeurig weergaveoverzichtobject dat onderdeel is van de gebeurtenisstroom van een gebeurtenisobject.

Code voor gebeurtenislisteners volgt altijd de volgende standaardstructuur (elementen die vet zijn weergegeven, moet u voor uw eigen, specifieke geval invullen):

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

In deze code gebeuren twee dingen. Eerst wordt een functie gedefinieerd, waarin de handelingen worden opgegeven die als reactie op de gebeurtenis worden uitgevoerd. Vervolgens wordt de methode `addEventListener()` van het bronobject aangeroepen, waarmee in wezen de functie wordt 'geabonneerd' op de opgegeven gebeurtenis, zodat wanneer deze gebeurtenis plaatsvindt, de handelingen van de functie worden uitgevoerd. Wanneer de gebeurtenis feitelijk plaatsvindt, controleert het gebeurtenisdoel het overzicht met alle functies en methoden die als gebeurtenislistener zijn geregistreerd. Vervolgens worden achtereenvolgens alle functies en methoden opgeroepen, waarbij het gebeurtenisobject als parameter wordt doorgegeven.

Als u een eigen gebeurtenislistener wilt maken, moet u vier dingen in deze code wijzigen. Ten eerste moet u de naam van de functie wijzigen in de naam die u wilt gebruiken (dit moet tweemaal gebeuren, op de plaats waar **eventResponse**) staat. Vervolgens moet u de juiste klassennaam opgeven van het gebeurtenisobject dat wordt verzonden naar aanleiding van de gebeurtenis waarnaar u wilt luisteren (**EventType** in de code) en moet u de juiste constante voor de specifieke gebeurtenis opgeven (**EVENT_NAME** in het overzicht). Vervolgens moet u de methode `addEventListener()` aanroepen op het object waardoor de gebeurtenis wordt verzonden (**eventTarget** in deze code). U kunt ook de naam wijzigen van de variabele die als parameter van de functie wordt gebruikt (**eventObject** in deze code).

Belangrijke concepten en termen

De volgende lijst bevat belangrijke termen die u tijdens het schrijven van gebeurtenisafhandelingsroutines zult tegenkomen:

Bubbling Bubbling vindt plaats voor bepaalde gebeurtenissen, zodat een bovenliggend weergaveobject kan reageren op gebeurtenissen die door de onderliggende objecten worden verzonden.

Bubblingfase Het gedeelte van de gebeurtenisstroom waarin een gebeurtenis aan bovenliggende weergaveobjecten wordt doorgegeven. De bubblingfase vindt plaats na de vastleg- en doelfasen.

Vastlegfase Het gedeelte van de gebeurtenisstroom waarin een gebeurtenis van het meest algemene doel aan het meest specifieke doelobject wordt doorgegeven. De vastlegfase vindt plaats voor de doel- en bubblingfasen.

Standaardgedrag Bij sommige gebeurtenissen hoort gedrag dat normaal gesproken samen met de gebeurtenis optreedt. Dit wordt het standaardgedrag genoemd. Als een gebruiker bijvoorbeeld tekst in een tekstveld typt, treedt een tekstinvoergebeurtenis op. Het standaardgedrag voor die gebeurtenis is het feitelijke weergeven van het teken dat in het tekstveld was getypt. U kunt dit standaardgedrag echter wijzigen (als u om een of andere reden niet wilt dat het getypte teken wordt weergegeven).

Verzenden Gebeurtenislisteners een melding geven dat een gebeurtenis heeft plaatsgevonden.

Gebeurtenis Iets dat met een object gebeurt dat door dit object aan andere objecten kan worden doorgegeven.

Gebeurtenisstroom Wanneer een gebeurtenis plaatsvindt die betrekking heeft op een object in het weergaveoverzicht (een object dat op het scherm wordt weergegeven), krijgen alle objecten die het betreffende object bevatten hiervan een melding. Hierna wordt door deze objecten een melding verzonden naar de bijbehorende gebeurtenislisteners. Dit proces begint in het werkgebied, waar het weergaveoverzicht wordt doorlopen tot het object wordt bereikt waarop de gebeurtenis betrekking had. Hierna wordt naar het werkgebied teruggekeerd. Dit proces wordt de gebeurtenisstroom genoemd.

Event-object Een object dat informatie over een specifieke gebeurtenis bevat, dat bij het verzenden van de gebeurtenis naar alle listeners wordt verstuurd.

Gebeurtenisdoel Het object waardoor een gebeurtenis daadwerkelijk wordt verzonden. Als de gebruiker bijvoorbeeld op een knop klikt die zich in een Sprite bevindt die zich op zijn beurt in het werkgebied bevindt, worden door al die objecten gebeurtenissen verzonden, maar is het gebeurtenisdoel gelijk aan het object waarop de gebeurtenis betrekking had, in dit geval de knop waarop werd geklikt.

Listener Een object dat of een functie die zichzelf bij een object heeft geregistreerd, om aan te geven dat het op de hoogte moet worden gesteld wanneer een specifieke gebeurtenis plaatsvindt.

Doelfase Het punt in de gebeurtenisstroom waarop een gebeurtenis het meest specifieke doel heeft bereikt. De doelfase vindt plaats tussen de vastleggings- en bubblingfasen.

De verschillen in gebeurtenisafhandeling in ActionScript 3.0 ten opzichte van eerdere versies

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het meest opvallende verschil tussen gebeurtenisafhandeling in ActionScript 3.0 ten opzichte van eerdere versies van ActionScript is dat er in ActionScript 3.0 slechts één systeem voor gebeurtenisafhandeling is, terwijl er in eerdere versies van ActionScript verschillende systemen voor gebeurtenisafhandeling zijn. In deze sectie wordt eerst een overzicht gegeven van hoe gebeurtenisafhandeling werkt in eerdere versies van ActionScript en wordt vervolgens besproken hoe gebeurtenisafhandeling is gewijzigd in ActionScript 3.0.

Gebeurtenisafhandeling in eerdere versies van ActionScript

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Versies van ActionScript vóór ActionScript 3.0 kennen een aantal verschillende manieren om gebeurtenissen af te handelen:

- Gebeurtenishandlers `on()` die direct op instanties Button en MovieClip kunnen worden geplaatst
- Handlers `onClipEvent()` die direct op instanties MovieClip kunnen worden geplaatst
- Eigenschappen van callback-functies, zoals `XML.onload` en `Camera.onActivity`
- Gebeurtenislisteners die u registreert met de methode `addListener()`
- De klasse UIEventDispatcher waarbij het DOM-gebeurtenismodel gedeeltelijk is geïmplementeerd

Al deze mechanismen kennen hun eigen voordelen en beperkingen. De handlers `on()` en `onClipEvent()` zijn eenvoudig te gebruiken. Het onderhoud van projecten is echter moeilijker, omdat code die direct op knoppen en filmclips is geplaatst, vaak lastig te vinden is. Callback-functies zijn ook eenvoudig te implementeren, maar u kunt slechts één callback-functie per gebeurtenis gebruiken. Gebeurtenislisteners zijn moeilijker te implementeren. U moet niet alleen een listenerobject en -functie maken, maar ook de listener registreren bij het object waardoor de gebeurtenis wordt gegenereerd. Door deze verhoogde overhead kunt u echter wel verschillende listenerobjecten maken en deze allemaal voor dezelfde gebeurtenis registreren.

Bij de ontwikkeling van componenten voor ActionScript 2.0 is een ander gebeurtenismodel ontstaan. Dit nieuwe model, dat in de klasse UIEventDispatcher is opgenomen, was gebaseerd op een subset van de DOM Events Specification. Voor ontwikkelaars die bekend zijn met gebeurtenisafhandeling voor componenten, is de overgang naar het nieuwe ActionScript 3.0-gebeurtenismodel relatief eenvoudig.

Helaas kent de syntaxis die door de verschillende gebeurtenismodellen wordt gebruikt, zowel overlap als verschillen. In ActionScript 2.0 kunnen sommige eigenschappen, zoals `TextField.onChanged`, als callback-functie maar ook als gebeurtenislistener worden gebruikt. Voor de syntaxis voor het registreren van listenerobjecten is het echter van belang of u een van de zes klassen gebruikt die ondersteuning bieden voor listeners, of de klasse UIEventDispatcher. Voor de klassen Key, Mouse, MovieClipLoader, Selection, Stage en TextField kunt u de methode `addListener()` gebruiken, maar voor gebeurtenisafhandeling voor componenten gebruikt u de methode `addEventListener()`.

Een andere complicatie bij het gebruik van verschillende gebeurtenisafhandelingsmodellen is dat het bereik van de gebeurtenishandlerfunctie veel kan verschillen, afhankelijk van het gebruikte mechanisme. Met andere woorden, de betekenis van het trefwoord `this` is niet consistent tussen de systemen voor gebeurtenisafhandeling.

Gebeurtenisafhandeling in ActionScript 3.0

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 introduceert één gebeurtenisafhandelingsmodel ter vervanging van de vele verschillende gebeurtenisafhandelingsmechanismen die in vorige versies van ActionScript bestaan. Het nieuwe gebeurtenismodel is gebaseerd op de DOM (Document Object Model) Level 3 Events Specification. Hoewel de bestandsindeling SWF niet specifiek voldoet aan de DOM-standaard, bestaan er voldoende overeenkomsten tussen het weergaveoverzicht en de DOM-structuur om implementatie van het DOM-gebeurtenismodel mogelijk te maken. In de hiërarchische DOM-structuur is een object in het weergaveoverzicht gelijk aan een node en de termen *weergaveoverzichtobject* en *node* worden hier door elkaar gebruikt.

De Flash Player- en AIR-implementatie van het DOM-gebeurtenismodel kent het concept standaardgedragingen. Een *standaardgedrag* is een handeling die door Flash Player of AIR wordt uitgevoerd als het normale gevolg van een bepaalde gebeurtenis.

Standaardgedrag

Code om op gebeurtenissen te reageren wordt meestal door ontwikkelaars geschreven. In sommige gevallen wordt gedrag echter zo vaak geassocieerd met een gebeurtenis dat dit gedrag automatisch door Flash Player of AIR wordt uitgevoerd, tenzij de ontwikkelaar code toevoegt om dit te annuleren. Aangezien dergelijk gedrag automatisch door Flash Player of AIR wordt uitgevoerd, worden deze gedragingen standaardgedragingen genoemd.

Als een gebruiker bijvoorbeeld tekst in een object TextField invoert, is de verwachting dat de tekst in dat TextField-object wordt weergegeven zó standaard dat het gedrag in Flash Player en AIR is ingebouwd. Als u niet wilt dat dit standaardgedrag optreedt, kunt u dit met het nieuwe systeem voor gebeurtenisafhandeling annuleren. Als een gebruiker tekst in een TextField-object invoert, wordt door Flash Player of AIR een instantie van de klasse TextEvent gemaakt die deze gebruikersinvoer vertegenwoordigt. Als u wilt voorkomen dat Flash Player of AIR de tekst in het TextField-object weergeeft, moet u die specifieke TextEvent-instantie benaderen en de methode `preventDefault()` van die instantie aanroepen.

Niet al het standaardgedrag kan worden voorkomen. Als een gebruiker bijvoorbeeld dubbelklikt op een woord in een TextField-object, wordt door Flash Player en AIR een MouseEvent-object gegenereerd. Het standaardgedrag, dat niet kan worden voorkomen, is dat het woord onder de cursor wordt gemarkeerd.

Aan een groot aantal gebeurtenisobjecttypen is geen standaardgedrag gekoppeld. Flash Player verzendt bijvoorbeeld een gebeurtenisobject connect wanneer een netwerkverbinding tot stand is gebracht, maar hieraan is geen standaardgedrag gekoppeld. In de API-documentatie voor de klasse Event en de subklassen hiervan worden alle gebeurtenistypen en al het hieraan gekoppelde standaardgedrag beschreven. In de documentatie wordt ook aangegeven of dat gedrag kan worden voorkomen.

Het is belangrijk om te begrijpen dat standaardgedragingen alleen gekoppeld kunnen zijn aan gebeurtenisobjecten die door Flash Player of AIR worden verzonden en dat deze niet bestaan voor gebeurtenisobjecten die via programmacode door ActionScript worden verzonden. U kunt bijvoorbeeld de methoden van de klasse EventDispatcher gebruiken om een gebeurtenisobject met type `textInput` te verzenden, maar aan dat gebeurtenisobject is geen standaardgedrag gekoppeld. Met andere woorden, Flash Player en AIR geven geen teken in een object TextField weer als resultaat van een gebeurtenis `textInput` die u via programmacode hebt verzonden.

Nieuw voor gebeurtenislisteners in ActionScript 3.0

Voor ervaren ontwikkelaars die de methode `addListener()` in ActionScript 2.0 gebruiken, kan het handig zijn de verschillen aan te geven tussen het gebeurtenislistenermodel van ActionScript 2.0 en het gebeurtenismodel van ActionScript 3.0. In de volgende lijst worden enkele belangrijke verschillen tussen de twee gebeurtenismodellen beschreven:

- U kunt gebeurtenislisteners in ActionScript 2.0 in sommige gevallen met `addListener()` en in andere gevallen met `addEventListener()` toevoegen, terwijl u in ActionScript 3.0 in alle gevallen `addEventListener()` gebruikt.
- ActionScript 2.0 kent geen gebeurtenisstroom, dus de methode `addListener()` kan alleen worden aangeroepen op het object dat de gebeurtenis uitzendt, terwijl in ActionScript 3.0 de methode `addEventListener()` op elk object dat deel uitmaakt van de gebeurtenisstroom kan worden aangeroepen.
- In ActionScript 2.0 kunnen gebeurtenislisteners functies, methoden of objecten zijn, terwijl in ActionScript 3.0 alleen functies of methoden gebeurtenislisteners kunnen zijn.

De gebeurtenisstroom

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

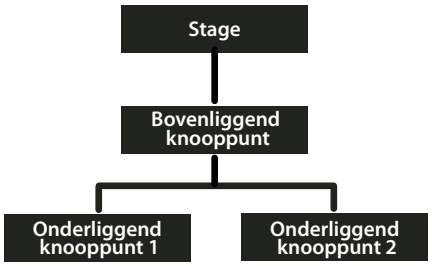
Flash Player of AIR verzendt gebeurtenisobjecten telkens wanneer een gebeurtenis plaatsvindt. Als het gebeurtenisdoel zich niet in het weergaveoverzicht bevindt, wordt het gebeurtenisobject direct door Flash Player of AIR naar het gebeurtenisdoel verzonden. Flash Player verzendt het gebeurtenisobject progress bijvoorbeeld direct naar een URLStream-object. Als het gebeurtenisdoel zich echter wel in het weergaveoverzicht bevindt, wordt het gebeurtenisobject door Flash Player naar het weergaveoverzicht verzonden en doorloopt het gebeurtenisobject het weergaveoverzicht tot aan het gebeurtenisdoel.

De *gebeurtenisstroom* beschrijft hoe een gebeurtenisobject het weergaveoverzicht doorloopt. Het weergaveoverzicht is geordend in een hiërarchie die als een boom kan worden beschreven. Aan de bovenkant in de hiërarchie bevindt zich het werkgebied, wat een speciale weergaveobjectcontainer is die als basis van het weergaveoverzicht fungeert. Het werkgebied wordt vertegenwoordigd door de klasse flash.display.Stage en kan alleen via een weergaveobject worden benaderd. Elk weergaveobject heeft een eigenschap met de naam `stage` die naar het werkgebied voor die toepassing verwijst.

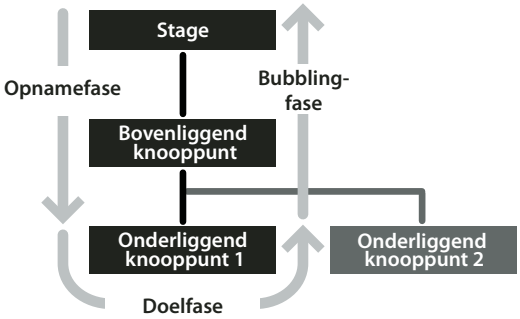
Wanneer Flash Player of AIR een gebeurtenisobject verstuurd voor een lijstgerelateerde weergavegebeurtenis, dan wordt dit gebeurtenisobject teruggestuurd vanaf de Stage naar het *doelknooppunt*. In de DOM Events Specification wordt de doelnodegedefinieerd als de node die het gebeurtenisdoel vertegenwoordigt. Met andere woorden, de doelnode is het weergaveoverzichtobject waar de gebeurtenis heeft plaatsgevonden. Als een gebruiker bijvoorbeeld op een weergaveoverzichtobject met de naam `child1` klikt, verzendt Flash Player of AIR een gebeurtenisobject waarbij `child1` de doelnode is.

De gebeurtenisstroom kan in drie fasen worden onderverdeeld. De eerste fase wordt de vastlegfase genoemd. Deze fase omvat alle nodes van het werkgebied tot aan de bovenliggende node van de doelnode. De tweede fase wordt de doelfase genoemd en bestaat uitsluitend uit de doelnode. De derde fase wordt de terugkoppelfase genoemd. Deze fase bestaat uit de nodes die worden aangetroffen op de terugtocht van de bovenliggende node van de doelnode naar het werkgebied.

De namen van de fasen zijn gemakkelijker te onthouden als u zich het weergaveoverzicht voorstelt als een verticale hiërarchie met aan de bovenkant het werkgebied, zoals in het volgende voorbeeld wordt getoond:



Als een gebruiker op `Child1` klikt, verzendt Flash Player of AIR een gebeurtenisobject naar de gebeurtenisstroom. Als de volgende afbeelding wordt weergegeven, begint de tocht van het object bij `Stage` en gaat deze vervolgens via `Parent Node` naar `Child1 Node`, om daarna weer terug te keren naar `Stage`. De terugtocht gaat via `Parent Node` naar `Stage`.



In dit voorbeeld bestaat de vastlegfase uit `Stage` en `Parent Node tijdens` de tocht omlaag. De doelfase bestaat uit `Child1 Node`. De terugkoppelfase bestaat uit `Parent Node` en `Stage` die tijdens de terugtocht weer voorbij komen.

De gebeurtenisstroom draagt bij aan een krachtiger systeem voor gebeurtenisafhandeling dan eerder beschikbaar was voor ActionScript-programmeurs. In eerdere versies van ActionScript bestaat de gebeurtenisstroom niet. Dit betekent dat gebeurtenislisteners alleen kunnen worden toegevoegd aan het object waardoor de gebeurtenis wordt gegenereerd. In ActionScript 3.0 kunt u gebeurtenislistener niet alleen aan een doelnode toevoegen, maar aan elke willekeurige node in de gebeurtenisstroom.

De mogelijkheid om gebeurtenislisteners in de gebeurtenisstroom toe te voegen is handig wanneer een gebruikersinterfacecomponent uit meerdere objecten bestaat. Een knopobject bevat bijvoorbeeld vaak een tekstobject dat als label voor de knop fungeert. Zonder de mogelijkheid om een listener aan de gebeurtenisstroom toe te voegen, moet u een listener aan zowel het knopobject als het tekstobject toevoegen, om er zeker van te zijn dat u een melding ontvangt wanneer er klikgebeurtenissen op een willekeurige locatie op de knop plaatsvinden. Dankzij de gebeurtenisstroom kunt u echter één gebeurtenislistener op het knopobject plaatsen, die zowel klikgebeurtenissen afhandelt die op het tekstobject plaatsvinden, als klikgebeurtenissen die plaatsvinden op plaatsen van de knop die niet achter het tekstobject zijn verborgen.

Niet elk gebeurtenisobject neemt echter deel aan alle drie fasen van de gebeurtenisstroom. Sommige gebeurtenistypen, zoals `enterFrame` en `init`, worden direct naar de doelnode verzonden en nemen noch aan de vastlegfase noch aan de terugkoppelfase deel. Andere gebeurtenissen kunnen als doel een object hebben dat zich niet in het weergaveoverzicht bevindt, zoals gebeurtenissen die naar een instantie van de klasse Socket worden verzonden. Deze gebeurtenisobjecten gaan ook direct naar het doelobject, zonder aan de vastleg- en terugkoppelfase deel te nemen.

Als u wilt weten hoe een bepaald gebeurtenistype zich gedraagt, raadpleegt u de API-documentatie of vraagt u de waarde op van de eigenschappen van het gebeurtenisobject. In de volgende sectie wordt beschreven hoe u de waarden van de eigenschappen van het gebeurtenisobject kunt opvragen.

Gebeurtenisobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebeurtenisobjecten hebben twee hoofddoelen in het nieuwe systeem voor gebeurtenisafhandeling. Het eerste is dat gebeurtenisobjecten feitelijke gebeurtenissen vertegenwoordigen door informatie over specifieke gebeurtenissen in een set eigenschappen op te slaan. Het tweede is dat gebeurtenisobjecten een set methoden bevatten waarmee u gebeurtenisobjecten kunt bewerken en het gedrag van het systeem voor gebeurtenisafhandeling kunt beïnvloeden.

Voor het mogelijk maken van toegang tot deze eigenschappen en methoden definieert de Flash Player-API een klasse Event die als basisklasse voor alle gebeurtenisobjecten fungeert. De klasse Event definieert een basisset eigenschappen en methoden die voor alle gebeurtenisobjecten van toepassing zijn.

In deze sectie worden eerst de eigenschappen en vervolgens de methoden van de klasse Event besproken. Ten slotte wordt uitgelegd waarom er subklassen van de klasse Event bestaan.

Eigenschappen van de klasse Event

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Event definieert een aantal alleen-lezen-eigenschappen en constanten die belangrijke informatie over een gebeurtenisobject bevatten. Hiervan zijn met name de volgende belangrijk:

- Gebeurtenisobjecttypen worden als constanten voorgesteld en in de eigenschap `Event.type` opgeslagen.
- Een Booleaanse waarde geeft aan of u het standaardgedrag van een gebeurtenis kunt voorkomen. Deze waarde wordt opgeslagen in de eigenschap `Event.cancelable`.

- In de resterende eigenschappen wordt informatie over de gebeurtenisstroom opgeslagen.

Gebeurtenisobjecttypen

Aan elk gebeurtenisobject is een gebeurtenistype gekoppeld. Gebeurtenistypen worden als tekenreekswaarden opgeslagen in de eigenschap `Event.type`. Het is handig om het type van een gebeurtenisobject te weten, zodat de code onderscheid kan maken tussen objecten van verschillende typen. De volgende code geeft bijvoorbeeld aan dat de listenerfunctie `clickHandler()` moet reageren op elk muisklikgebeurtenisobject dat aan `myDisplayObject` wordt doorgegeven:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

Er zijn meer dan twintig gebeurtenistypen gekoppeld aan de klasse Event zelf en deze worden vertegenwoordigd door constanten van de klasse Event. Enkele hiervan worden getoond in het volgende fragment van de definitie van de klasse Event:

```
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
    }
}
```

Met deze constanten kunt u op eenvoudige wijze verwijzen naar specifieke gebeurtenistypen. U moet deze constanten gebruiken in plaats van de tekenreeksen die deze vertegenwoordigen. Als de naam van een constante onjuist wordt gespeld in de code, wordt de fout door de compiler afgevangen. Als u echter tekenreeksen gebruikt, wordt een typefout mogelijk niet herkend bij compilatie, wat kan leiden tot onverwacht gedrag en fouten die moeilijk kunnen worden opgespoord. Bij het toevoegen van een gebeurtenislistener gebruikt u bijvoorbeeld de volgende code:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

en niet:

```
myDisplayObject.addEventListener("click", clickHandler);
```

Informatie over standaardgedrag

Door in de code de waarde van de eigenschap `cancelable` op te vragen, kunt u nagaan of standaardgedrag voor een bepaald gebeurtenisobject kan worden voorkomen. De eigenschap `cancelable` heeft een Booleaanse waarde die aangeeft of standaardgedrag kan worden voorkomen. Met de methode `preventDefault()` kunt u standaardgedrag dat aan een klein aantal gebeurtenissen is gekoppeld, voorkomen of annuleren. Zie Het standaardgedrag van gebeurtenissen annuleren onder “Methoden van de klasse Event” op pagina 142 voor meer informatie.

Informatie over de gebeurtenisstroom

De resterende eigenschappen van de klasse Event bevatten belangrijke informatie over een gebeurtenisobject en de relatie hiervan tot de gebeurtenisstroom, zoals in de volgende lijst wordt beschreven:

- De eigenschap `bubbles` bevat informatie over de fasen van de gebeurtenisstroom waaraan het gebeurtenisobject deelneemt.
- De eigenschap `eventPhase` geeft aan wat de huidige fase in de gebeurtenisstroom is.
- De eigenschap `target` bevat een verwijzing naar het gebeurtenisdoel.

- De eigenschap `currentTarget` bevat een verwijzing naar het weergaveoverzichtobject dat momenteel het gebeurtenisobject verwerkt.

De eigenschap bubbles

Een gebeurtenis loopt omhoog (bubble in het Engels) als het gebeurtenisobject aan de terugkoppelfase van de gebeurtenisstroom deelneemt. Dit houdt in dat het gebeurtenisobject vanuit de doelnode en via de voorouders naar het werkgebied terugkeert. De eigenschap `Event.bubbles` bevat een Booleaanse waarde die aangeeft of het gebeurtenisobject aan de terugkoppelfase deelneemt. Aangezien alle gebeurtenissen in de terugkoppelfase ook aan de vastleg- en doelfase deelnemen, nemen deze gebeurtenissen aan alle drie fasen van de gebeurtenisstroom deel. Als de waarde `true` is, neemt het gebeurtenisobject aan alle drie de fasen deel. Als de waarde `false` is, neemt het gebeurtenisobject niet deel aan de terugkoppelfase.

De eigenschap eventPhase

U kunt de gebeurtenisfase van elk gebeurtenisobject bepalen aan de hand van de waarde van de eigenschap `eventPhase`. De eigenschap `eventPhase` bevat een geheel-getalwaarde zonder teken die een van de drie fasen van de gebeurtenisstroom voorstelt. De Flash Player-API definieert een aparte klasse EventPhase die drie constanten bevat die overeenkomen met de drie geheel-getalwaarden zonder teken, zoals in het volgende codefragment wordt getoond:

```
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;
    }
}
```

Deze constanten komen overeen met de drie geldige waarden van de eigenschap `eventPhase`. U kunt deze constanten gebruiken om de code beter leesbaar te maken. Als u er bijvoorbeeld voor wilt zorgen dat een functie met de naam `myFunc()` alleen wordt aangeroepen als het gebeurtenisdoel zich in de doelfase bevindt, kunt u de volgende code gebruiken om op deze voorwaarde te testen:

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

De eigenschap target

De eigenschap `target` bevat een verwijzing naar het object dat het doel van de gebeurtenis is. In sommige gevallen is dit vrij duidelijk. Als een microfoon actief wordt, is het doel van het gebeurtenisobject bijvoorbeeld het microfoonobject. Als het doel zich echter in het weergaveoverzicht bevindt, moet u rekening houden met de hiërarchie van het weergaveoverzicht. Als een gebruiker bijvoorbeeld een muisklik uitvoert op een punt met overlappende weergaveoverzichtobjecten, kiezen Flash Player en AIR altijd het object dat het verst verwijderd is van het werkgebied als gebeurtenisdoel.

Voor veel complexe SWF-bestanden, vooral die waarbij zich veel kleinere onderliggende objecten op knoppen bevinden, kunt u de eigenschap `target` niet gebruiken, omdat deze naar een onderliggend object van een knop verwijst in plaats van naar de knop zelf. In die gevallen is het gebruikelijk om gebeurtenislisteners toe te voegen aan de knop en de eigenschap `currentTarget` te gebruiken, omdat deze wel naar de knop verwijst, terwijl de eigenschap `target` naar een onderliggend object van de knop kan verwijzen.

De eigenschap currentTarget

De eigenschap `currentTarget` bevat een verwijzing naar het object dat momenteel het gebeurtenisobject verwerkt. Hoewel het vreemd kan lijken om niet te weten welke node momenteel het door u bestudeerde gebeurtenisobject verwerkt, moet u niet vergeten dat u een listenerfunctie kunt toevoegen aan elk weergaveobject in de gebeurtenisstroom van dat gebeurtenisobject en dat de listenerfunctie op elke locatie kan worden geplaatst. Verder kan een bepaalde listenerfunctie aan verschillende weergaveobjecten worden toegevoegd. Hoe groter en complexer een project is, des te nuttiger de eigenschap `currentTarget` is.

Methoden van de klasse Event

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er bestaan drie categorieën methoden van de klasse Event:

- Hulpprogrammamethoden, waarmee u kopieën van een gebeurtenisobject kunt maken of dit in een tekenreeks kunt omzetten
- Gebeurtenisstroommethoden, waarmee u gebeurtenisobjecten uit de gebeurtenisstroom kunt verwijderen
- Standaardgedragsmethoden, die standaardgedrag voorkomen of controleren of dit is voorkomen

Hulpprogrammamethoden van de klasse Event

De klasse Event kent twee hulpprogrammamethoden. Met de methode `clone()` kunt u kopieën van een gebeurtenisobject maken. Met de methode `toString()` kunt u een tekenreeksrepresentatie genereren van de eigenschappen van een gebeurtenisobject en de waarden hiervan. Beide methoden worden intern gebruikt door het gebeurtenismodelsysteem, maar worden voor algemeen gebruik aan ontwikkelaars aangeboden.

Voor gevorderde ontwikkelaars die subklassen van de klasse Event maken, moet u van beide hulpprogrammamethoden een versie overschrijven en implementeren om te zorgen dat de Event-subklasse correct werkt.

Gebeurtenisstroom stoppen

U kunt de methode `Event.stopPropagation()` of de methode `Event.stopImmediatePropagation()` aanroepen om te zorgen dat de tocht van een gebeurtenisobject door de gebeurtenisstroom wordt afgebroken. De twee methoden zijn bijna identiek en verschillen in het feit of de andere gebeurtenislisteners van de huidige node al dan niet mogen worden uitgevoerd:

- De methode `Event.stopPropagation()` voorkomt dat het gebeurtenisobject naar de volgende node doorgaat, maar pas nadat eventuele andere gebeurtenislisteners op de huidige node zijn uitgevoerd.
- De methode `Event.stopImmediatePropagation()` voorkomt ook dat het gebeurtenisobject naar de volgende node doorgaat, maar in dit geval mogen geen andere gebeurtenislisteners op de huidige node meer worden uitgevoerd.

Het aanroepen van deze methoden heeft geen effect op het feit of het standaardgedrag dat aan een gebeurtenis is gekoppeld, wordt uitgevoerd. Voor het voorkomen van standaardgedrag gebruikt u de standaardgedragsmethoden van de klasse Event.

Standaardgedrag voor een gebeurtenis annuleren

De twee methoden waarmee standaardgedrag kan worden geannuleerd, zijn `preventDefault()` en `isDefaultPrevented()`. U roept de methode `preventDefault()` aan om het standaardgedrag te annuleren dat aan een gebeurtenis is gekoppeld. Als u wilt controleren of `preventDefault()` al op een gebeurtenisobject is aangeroepen, roept u de methode `isDefaultPrevented()` aan, die de waarde `true` retourneert als de methode al is aangeroepen en `false` als dat niet het geval is.

De methode `preventDefault()` werkt alleen als het standaardgedrag van een gebeurtenis kan worden geannuleerd. U kunt controleren of dit het geval is door de API-documentatie voor dat gebeurtenistype te raadplegen of door met behulp van ActionScript de waarde van de eigenschap `cancelable` van het gebeurtenisobject op te vragen.

Als u het standaardgedrag annuleert, heeft dit geen effect op de voortgang van een gebeurtenisobject in de gebeurtenisstroom. Als u een gebeurtenisobject uit de gebeurtenisstroom wilt verwijderen, gebruikt u de gebeurtenisstroommethoden van de klasse Event.

Subklassen van de klasse Event

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voor veel gebeurtenissen is de in de klasse Event gedefinieerde standaardset van eigenschappen voldoende. Andere gebeurtenissen hebben echter unieke kenmerken die niet kunnen worden vastgelegd met de in de klasse Event gedefinieerde eigenschappen. Voor deze gebeurtenissen zijn in ActionScript 3.0 een aantal subklassen van de klasse Event gedefinieerd.

Elke subklasse bevat aanvullende eigenschappen en gebeurtenistypen die uniek zijn voor die gebeurteniscategorie. Gebeurtenissen die te maken hebben met muisinvoer, hebben bijvoorbeeld diverse unieke kenmerken die niet kunnen worden vastgelegd met de in de klasse Event gedefinieerde eigenschappen. De klasse MouseEvent breidt de klasse Event uit door tien eigenschappen toe te voegen die bepaalde informatie bevatten, zoals de locatie van de muisgebeurtenis en of bepaalde toetsen zijn ingedrukt tijdens de muisgebeurtenis.

Een subklasse van de klasse Event bevat ook constanten die de gebeurtenistypen vertegenwoordigen die aan de subklasse zijn gekoppeld. De klasse MouseEvent definieert bijvoorbeeld constanten voor verschillende muisgebeurtenistypen, waaronder de gebeurtenistypen `click`, `doubleClick`, `mouseDown` en `mouseUp`.

Zoals in de sectie over hulpprogrammamethoden van de klasse Event onder “Gebeurtenisobjecten” op pagina 139 is beschreven, moet u bij het maken van een subklasse van de klasse Event de methoden `clone()` en `toString()` opheffen om functionaliteit te bieden die specifiek is voor de subklasse.

Gebeurtenislisteners

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebeurtenislisteners, die ook gebeurtenishandlers worden genoemd, zijn functies die door Flash Player en AIR worden uitgevoerd als reactie op specifieke gebeurtenissen. Het toevoegen van een gebeurtenislistener bestaat uit twee stappen. Eerst maakt u een functie of een klassenmethode die door Flash Player of AIR kan worden uitgevoerd als reactie op de gebeurtenis. Dit wordt soms de listenerfunctie of gebeurtenishandlerfunctie genoemd. Vervolgens gebruikt u de methode `addEventListener()` om de listenerfunctie te registreren bij het doel van de gebeurtenis of bij een willekeurig weergaveoverzichtobject dat in de juiste gebeurtenisstroom ligt.

Een listenerfunctie maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het maken van listenerfuncties is één gebied waarop het ActionScript 3.0-gebeurtenismodel afwijkt van het DOM-gebeurtenismodel. In het DOM-gebeurtenismodel is een duidelijk verschil tussen een gebeurtenislistener en een listenerfunctie: een gebeurtenislistener is een instantie van een klasse die de interface EventListener implementeert, terwijl een listenerfunctie een methode van die klasse met de naam `handleEvent()` is. In het DOM-gebeurtenismodel registreert u de klasseninstantie die de listenerfunctie bevat en niet de eigenlijke listenerfunctie.

In het ActionScript 3.0-gebeurtenismodel bestaat er geen onderscheid tussen een gebeurtenislistener en een listenerfunctie: ActionScript 3.0 kent geen interface EventListener en listenerfuncties kunnen buiten een klasse of als deel van een klasse worden gedefinieerd. Verder hoeven listenerfuncties niet `handleEvent()` te worden genoemd. Elke geldige naam is toegestaan. In ActionScript 3.0 registreert u de naam van de feitelijke listenerfunctie.

Listenerfunctie die buiten een klasse is gedefinieerd

Met de volgende code wordt een eenvoudig SWF-bestand gemaakt waarmee een rood vierkant wordt weergegeven. Een listenerfunctie met de naam `clickHandler()`, die geen deel uitmaakt van een klasse, luistert naar muisklikgebeurtenissen op het rode vierkant.

```
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);
}
```

Wanneer een gebruiker met het resulterende SWF-bestand communiceert door op het vierkant te klikken, genereert Flash Player of AIR de volgende trace-uitvoer:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Het gebeurtenisobject wordt als argument doorgegeven aan `clickHandler()`. Hierdoor kan het gebeurtenisobject door de listenerfunctie worden gecontroleerd. In dit voorbeeld gebruikt u de eigenschap `type` van het gebeurtenisobject om te controleren of de gebeurtenis een klikgebeurtenis is.

In het voorbeeld wordt ook de waarde van het trefwoord `this` gecontroleerd. In dit geval vertegenwoordigt `this` het algemene object. Dit is logisch, omdat de functie buiten enige aangepaste klasse of enig aangepast object is gedefinieerd.

Listenerfunctie die als klassenmethode is gedefinieerd

Het volgende voorbeeld is gelijk aan het vorige ClickExample-voorbeeld, met het verschil dat de functie `clickHandler()` als een methode van de klasse ChildSprite is gedefinieerd:

```
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);
    }
}
```

Wanneer een gebruiker met het resulterende SWF-bestand communiceert door op het rode vierkant te klikken, genereert Flash Player of AIR de volgende trace-uitvoer:

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

Het trefwoord `this` verwijst naar de ChildSprite-instantie met de naam `child`. Dit gedrag wijkt af van het gedrag in ActionScript 2.0. Als u componenten gebruikte in ActionScript 2.0, weet u wellicht nog wel dat wanneer een klassenmethode werd doorgegeven aan `UIEventDispatcher.addEventListener()`, het bereik van de methode was gekoppeld aan de component waardoor de gebeurtenis werd verzonden en niet aan de klasse waarin de listenermethode werd gedefinieerd. Met andere woorden, als u deze techniek in ActionScript 2.0 gebruikte, verwees het trefwoord `this` naar de component die de gebeurtenis verzond en niet naar de ChildSprite-instantie.

Dit kon een serieus probleem voor sommige programmeurs zijn, omdat dit betekende dat deze geen toegang hadden tot andere methoden en eigenschappen van de klasse waarin de listenermethode werd gedefinieerd. Als oplossing konden ActionScript 2.0-programmeurs de klasse `mx.util.Delegate` gebruiken om het bereik van de listenermethode te wijzigen. Dit is echter niet meer nodig, omdat ActionScript 3.0 een gekoppelde methode maakt wanneer `addEventListener()` wordt aangeroepen. Als gevolg hiervan verwijst het trefwoord `this` naar de ChildSprite-instantie met de naam `child` en heeft de programmeur toegang tot de andere methoden en eigenschappen van de klasse ChildSprite.

Gebeurtenislistener waarvan het gebruik wordt afgeraden

Er bestaat een derde techniek waarbij u een algemeen object maakt met een eigenschap die verwijst naar een dynamisch toegewezen listenerfunctie. Het gebruik hiervan wordt echter niet aanbevolen. Deze techniek wordt hier besproken, omdat deze veel werd gebruikt in ActionScript 2.0, maar in ActionScript 3.0 beter niet gebruikt kan worden. Deze techniek wordt afgeraden omdat het trefwoord `this` naar het algemene object verwijst en niet naar het listenerobject.

Het volgende voorbeeld is gelijk aan het vorige ClickExample-voorbeeld, met het verschil dat de listenerfunctie is gedefinieerd als deel van een algemeen object met de naam `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);
}
```

De resultaten van de trace zien er als volgt uit:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Men zou verwachten dat `this` naar `myListenerObj` verwijst en dat de trace-uitvoer [`object Object`] is. Dit is echter niet het geval, this verwijst naar het algemene object. Wanneer u de naam van een dynamische eigenschap als argument doorgeeft aan `addEventListener()`, kan Flash Player of AIR geen gekoppelde methode maken. De reden hiervoor is dat u als parameter `listener` slechts het geheugenadres van de listenerfunctie doorgeeft en dat Flash Player en AIR dit geheugenadres op geen enkele manier kunnen koppelen aan de instantie `myListenerObj`.

Gebeurtenislisteners beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt listenerfuncties beheren met behulp van de methoden van de interface IEventDispatcher. De interface IEventDispatcher is de ActionScript 3.0-versie van de interface EventTarget van het DOM-gebeurtenismodel. Hoewel de naam IEventDispatcher lijkt te impliceren dat het hoofddoel het verzenden van gebeurtenisobjecten is, worden de methoden van deze klasse veel vaker gebruikt voor het registreren van, controleren op en verwijderen van gebeurtenislisteners. De interface IEventDispatcher definieert vijf methoden, zoals in de volgende code wordt weergegeven:

```
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;
    }
}
```

De Flash Player-API implementeert de interface IEventDispatcher met de klasse EventDispatcher, die als basisklasse fungeert voor alle klassen die gebeurtenisdoelen of onderdelen van een gebeurtenisstroom kunnen zijn. De klasse DisplayObject overerft bijvoorbeeld van de klasse EventDispatcher. Dit betekent dat elk object op het weergaveoverzicht toegang heeft tot de methoden van de interface IEventDispatcher.

Gebeurtenislisteners toevoegen

De methode `addEventListener()` is het werkpaard van de interface IEventDispatcher. U gebruikt deze voor het registreren van de listenerfuncties. De twee vereiste parameters zijn `type` en `listener`. U gebruikt de parameter `type` om het type gebeurtenis op te geven. U gebruikt de parameter `listener` om op te geven welke listenerfunctie wordt uitgevoerd wanneer de gebeurtenis plaatsvindt. De parameter `listener` kan een verwijzing zijn naar een functie of naar een klassenmethode.

Gebruik geen haakjes bij het opgeven van de parameter `listener`. De functie `clickHandler()` wordt bijvoorbeeld zonder haakjes opgegeven in de volgende aanroep van de methode `addEventListener()`:

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

Met de parameter `useCapture` van de methode `addEventListener()` kunt u controleren in welke fase van de gebeurtenisstroom de listener actief wordt. Als `useCapture` op `true` is ingesteld, wordt de listener actief tijdens de vastlegfase van de gebeurtenisstroom. Als `useCapture` op `false` is ingesteld, wordt de listener actief tijdens de doel- en terugkoppelfasen van de gebeurtenisstroom. Wanneer u in alle drie fasen naar de gebeurtenis wilt luisteren, roept u `addEventListener()` tweemaal aan, één keer met `useCapture` ingesteld op `true` en één keer met `useCapture` ingesteld op `false`.

De parameter `priority` van de methode `addEventListener()` maakt niet officieel deel uit van het DOM Level 3-gebeurtenismodel. Het is opgenomen in ActionScript 3.0 om u meer flexibiliteit te bieden bij het indelen van de gebeurtenislisteners. Wanneer u `addEventListener()` aanroept, kunt u de prioriteit voor die gebeurtenislistener instellen door een geheel-getalwaarde als parameter `priority` door te geven. De standaardwaarde is 0, maar u kunt deze wijzigen in een negatieve of positieve geheel-getalwaarde. Hoe hoger het getal, des te sneller die gebeurtenislistener wordt uitgevoerd. Gebeurtenislisteners met dezelfde prioriteit worden uitgevoerd in de volgorde waarin deze zijn toegevoegd. Hoe eerder een listener wordt toegevoegd, des te sneller deze wordt uitgevoerd.

Met de parameter `useWeakReference` kunt u opgeven of de verwijzing naar de listenerfunctie zwak of normaal is. Als u deze parameter op `true` instelt, kunt u situaties vermijden waarbij listenerfuncties in het geheugen aanwezig blijven, terwijl deze niet meer vereist zijn. Flash Player en AIR maken gebruik van een techniek voor het verwijderen van objecten uit het geheugen die niet meer in gebruik zijn. Deze techniek wordt *opschonen* genoemd. Een object is niet meer in gebruik als er geen verwijzingen naar bestaan. De opschoonfunctie negeert zwakke verwijzingen. Dit betekent dat als er alleen een zwakke verwijzing naar een listenerfunctie bestaat, de listenerfunctie in aanmerking komt voor opschonen.

Gebeurtenislisteners verwijderen

U kunt de methode `removeEventListener()` gebruiken om een gebeurtenislistener te verwijderen die u niet meer nodig hebt. Het is raadzaam om alle listeners die niet meer worden gebruikt, te verwijderen. Vereiste parameters zijn onder meer de parameters `eventName` en `listener`. Dit zijn dezelfde vereiste parameters als voor de methode `addEventListener()`. U kunt tijdens alle gebeurtenisfasen naar gebeurtenissen luisteren door `addEventListener()` tweemaal aan te roepen, één keer met `useCapture` ingesteld op `true` en één keer met deze parameter ingesteld op `false`. Als u beide gebeurtenislisteners wilt verwijderen, roept u `removeEventListener()` tweemaal aan, één keer met `useCapture` ingesteld op `true` en één keer met deze parameter ingesteld op `false`.

Gebeurtenissen verzenden

De methode `dispatchEvent()` kan door gevorderde programmeurs worden gebruikt om een aangepast gebeurtenisobject naar de gebeurtenisstroom te verzenden. De enige parameter die door deze methode wordt geaccepteerd, is een verwijzing naar een gebeurtenisobject, dat een instantie moet zijn van de klasse Event of van een subklasse van de klasse Event. Als de gebeurtenis is verzonden, wordt de eigenschap `target` van het gebeurtenisobject ingesteld op het object waarop `dispatchEvent()` was aangeroepen.

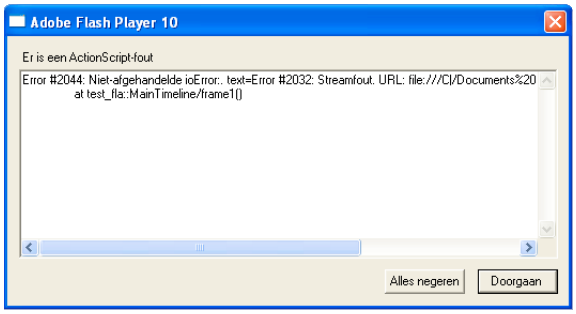
Controleren op bestaande gebeurtenislisteners

De laatste twee methoden van de interface IEventDispatcher bevatten nuttige informatie over het bestaan van gebeurtenislisteners. De methode `hasEventListener()` retourneert `true` als er een gebeurtenislistener is gevonden voor een specifiek gebeurtenistype op een bepaald weergaveoverzichtobject. De methode `willTrigger()` retourneert ook `true` als er een listener is gevonden voor een bepaald weergaveobject, maar `willTrigger()` zoekt niet alleen naar listeners op dat weergaveobject, maar ook naar listeners op alle voorouders van dat weergaveoverzichtobject voor alle fasen van de gebeurtenisstroom.

Foutgebeurtenissen zonder listeners

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uitzonderingen, en niet gebeurtenissen, vormen het belangrijkste mechanisme voor foutafhandeling in ActionScript 3.0. Foutafhandeling werkt echter niet voor asynchrone bewerkingen zoals het laden van bestanden. Als tijdens een dergelijke asynchrone bewerking een fout optreedt, verzenden Flash Player en AIR een foutgebeurtenisobject. Als u geen listener voor de foutgebeurtenis maakt, wordt door de foutopsporingsversies van Flash Player en AIR een dialoogvenster weergegeven met informatie over de fout. Zo genereert de foutopsporingsversie van Flash Player het volgende dialoogvenster waarin de fout wordt beschreven wanneer de toepassing een bestand vanaf een ongeldige URL probeert te laden:



De meeste foutgebeurtenissen zijn gebaseerd op de klasse ErrorEvent en hebben als zodanig een eigenschap met de naam `text`, waarmee het foutbericht wordt opgeslagen dat door Flash Player of AIR wordt weergegeven. De twee uitzonderingen zijn de klassen StatusEvent en NetStatusEvent. Deze klassen hebben een eigenschap `level` (`StatusEvent.level` en `NetStatusEvent.info.level`). Wanneer de waarde van de eigenschap `level` gelijk is aan "`error`", worden deze gebeurtenistypen als foutgebeurtenissen beschouwd.

Een foutgebeurtenis zorgt er niet voor dat de uitvoering van een SWF-bestand wordt afgesloten. Deze wordt alleen weergegeven als een dialoogvenster in de foutopsporingsversies van de insteekmodules van de browser en zelfstandige spelers, als een bericht in het uitvoervenster in de oorspronkelijke speler en als een item in het logbestand voor Adobe Flash Builder. In de releaseversies van Flash Player of AIR levert een foutgebeurtenis geen enkele melding op.

Voorbeeld van gebeurtenisafhandeling: alarmklok

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het wekkervoorbeeld bestaat uit een klok waarbij de gebruiker een tijdstip kan aangeven waarop de wekker afgaat en een bericht wordt weergegeven. Het wekkervoorbeeld is gebaseerd op de toepassing SimpleClock in “Werken met datums en tijden” op pagina 1. Met dit voorbeeld worden verschillende aspecten van het werken met gebeurtenissen in ActionScript 3.0 geïllustreerd, waaronder:

- Luisteren naar en reageren op een gebeurtenis
- Listeners een melding geven van een gebeurtenis
- Een aangepast gebeurtenistype maken

Zie http://www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de Flash Professional-toepassingsbestanden voor dit voorbeeld wilt downloaden. Zie http://www.adobe.com/go/as3examples_nl als u de Flex-toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de bestanden voor de wekkertoepassing in de map Samples/AlarmClock. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
AlarmClockApp.mxml of AlarmClockApp.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/clock/AlarmClock.as	Een klasse die de klasse SimpleClock uitbreidt en wekkerfunctionaliteit toevoegt.
com/example/programmingas3/clock/AlarmEvent.as	Een aangepaste Event-klasse (een subklasse van flash.events.Event), die als gebeurtenisobject fungeert voor de gebeurtenis `alarm` van de klasse AlarmClock.
com/example/programmingas3/clock/AnalogClockFace.as	Hiermee wordt een ronde wijzerplaat getekend met uur-, minuut- en secondewijzers die de tijd aangeven (beschreven in het voorbeeld SimpleClock).
com/example/programmingas3/clock/SimpleClock.as	Een wijzerplaatcomponent met eenvoudige tijdwaarnemingsfunctionaliteit (beschreven in het voorbeeld SimpleClock).

Overzicht wekker

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In dit voorbeeld wordt voor de hoofdfunctionaliteit van de klok, zoals het bijhouden van de tijd en het weergeven van de wijzerplaat, opnieuw gebruikgemaakt van de code van de toepassing SimpleClock, zoals beschreven in "Voorbeeld van datum en tijd: eenvoudige analoge klok" op pagina 6. De klasse AlarmClock breidt de klasse SimpleClock van dat voorbeeld uit door de functionaliteit toe te voegen die voor een wekker is vereist, zoals het instellen van de wektijd en het geven van een melding wanneer de wekker 'afgaat'.

Het geven van een melding wanneer er iets gebeurt is de taak waarvoor gebeurtenissen zijn bedoeld. Met de klasse AlarmClock wordt de gebeurtenis alarm beschikbaar gemaakt, waar andere objecten naar kunnen luisteren teneinde de juiste handelingen uit te voeren. Hiernaast maakt de klasse AlarmClock gebruik van een instantie van de klasse Timer om te bepalen wanneer de wekker moet afgaan. Net als de klasse AlarmClock kent de klasse Timer een gebeurtenis voor het doorgeven van meldingen aan andere objecten (een instantie AlarmClock in dit geval) wanneer een bepaalde hoeveelheid tijd is verstreken. Net als bij de meeste ActionScript-toepassingen maken bij het wekkervoorbeeld gebeurtenissen een belangrijk deel uit van de functionaliteit.

De wekker af laten gaan

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals eerder is vermeld, heeft de enige feitelijke functionaliteit die de klasse AlarmClock biedt, betrekking op het instellen en af laten gaan van de wekker. Met de ingebouwde klasse Timer (flash.utils.Timer) kunnen ontwikkelaars code schrijven die na een bepaalde hoeveelheid tijd wordt uitgevoerd. De klasse AlarmClock maakt gebruik van een instantie Timer om te bepalen wanneer de wekker moet afgaan.

```
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);
}
```

De in de klasse AlarmClock gedefinieerde Timer-instantie heeft de naam `alarmTimer`. De methode `initClock()`, die vereiste instelbewerkingen uitvoert voor de instantie AlarmClock, doet twee dingen met de variabele `alarmTimer`. Eerst wordt de variabele geïnstantieerd met parameters die de Timer-instantie de opdracht geven om 0 milliseconden te wachten en de wekkergebeurtenis slechts één keer te activeren. Nadat `alarmTimer` is geïnstantieerd, roept de code de methode `addEventListener()` van die variabele aan om aan te geven dat deze naar de gebeurtenis `timer` van die variabele wil luisteren. Een instantie Timer werkt door de gebeurtenis `timer` te verzenden nadat een bepaalde hoeveelheid tijd is verlopen. De klasse AlarmClock moet weten wanneer de gebeurtenis `timer` wordt verzonden voor het laten afgaan van de eigen wekker. Door `addEventListener()` aan te roepen, registreert de code van AlarmClock zichzelf als listener bij `alarmTimer`. De twee parameters geven aan dat de klasse AlarmClock naar de gebeurtenis `timer` wil luisteren (aangegeven door de constante `TimerEvent.TIMER`) en dat bij het plaatsvinden van de gebeurtenis de methode `onAlarm()` van de klasse AlarmClock moet worden aangeroepen als reactie op deze gebeurtenis.

Voor het feitelijke instellen van de wektijd wordt de methode `setAlarm()` van de klasse AlarmClock aangeroepen, zoals hier wordt aangegeven:

```
/**
 * 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;
}
```

Deze methode doet verschillende dingen, zoals het opslaan van het wekbericht en het maken van een Date-object (`alarmTime`), die het feitelijke moment vertegenwoordigt waarop de wekker moet afgaan. Het belangrijkste in deze code zijn hier de laatste paar regels van de methode, waar de wekker van de variabele `alarmTimer` wordt ingesteld en geactiveerd. Eerst wordt de methode `reset()` aangeroepen, waarbij de timer wordt gestopt en opnieuw wordt ingesteld, voor het geval deze al draait. Vervolgens wordt de huidige tijd (vertegenwoordigd door de variabele `now`) afgetrokken van de waarde van de variabele `alarmTime`, om te bepalen na hoeveel milliseconden de wekker kan afgaan. De gebeurtenis `timer` wordt niet op een absoluut tijdstip door de klasse Timer geactiveerd, zodat dit relatieve tijdsverschil wordt toegewezen aan de eigenschap `delay` van `alarmTimer`. Ten slotte wordt de methode `start()` aangeroepen om de timer daadwerkelijk te starten.

Als de opgegeven hoeveelheid tijd is verlopen, verzendt `alarmTimer` de gebeurtenis `timer`. Aangezien de klasse AlarmClock de methode `onAlarm()` als listener heeft geregistreerd voor die gebeurtenis, wordt bij het plaatsvinden van de gebeurtenis `timeronAlarm()` aangeroepen.

```
/**
 * 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);
}
```

Een methode die als gebeurtenislistener is geregistreerd, moet met de juiste handtekening worden gedefinieerd (dat wil zeggen, met de parameters en het retourneringstype die voor de methode gelden). Een methode is een listener voor de gebeurtenis `timer` van de klasse Timer als deze één parameter definieert waarvan het gegevenstype gelijk is aan TimerEvent (flash.events.TimerEvent), een subklasse van de klasse Event. Als de instantie Timer de gebeurtenislisteners aanroept, wordt een instantie TimerEvent als gebeurtenisobject doorgegeven.

Anderen op de hoogte stellen van het weksignaal

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Net als de klasse Timer kent de klasse AlarmClock een gebeurtenis die ervoor zorgt dat aan andere code meldingen kunnen worden gestuurd wanneer de wekker afgaat. Een klasse kan gebruikmaken van het systeem voor gebeurtenisafhandeling dat in ActionScript is ingebouwd, als deze klasse de interface flash.events.IEventDispatcher implementeert. Dit gebeurt meestal door de klasse flash.events.EventDispatcher uit te breiden, die een standaardimplementatie bevat van IEventDispatcher (of door een van de subklassen van EventDispatcher uit te breiden). Zoals hierboven werd beschreven, bevat de AlarmClock-klasse de SimpleClock-klasse, die (via een overervingsketen) de EventDispatcher-klasse bevat. Dit alles betekent dat de klasse AlarmClock al ingebouwde functionaliteit heeft om eigen gebeurtenissen aan te kunnen bieden.

Andere code kan zich registreren voor het ontvangen van een melding bij het plaatsvinden van de gebeurtenis `alarm` van de klasse AlarmClock door de methode `addEventListener()` aan te roepen die AlarmClock overerft van EventDispatcher. Wanneer een AlarmClock-instantie gereed is voor het verzenden van een melding aan andere code met betrekking tot het plaatsvinden van de gebeurtenis `alarm`, roept deze de methode `dispatchEvent()` aan, die ook wordt overgeërfd van EventDispatcher.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

Deze regels code zijn afkomstig van de methode `onAlarm()` van de klasse AlarmClock (die eerder in haar geheel is weergegeven). De methode `dispatchEvent()` van de AlarmClock-instantie wordt aangeroepen, waardoor weer een melding wordt verzonden naar alle geregistreerde listeners dat de gebeurtenis `alarm` van de AlarmClock-instantie heeft plaatsgevonden. De parameter die aan `dispatchEvent()` wordt doorgegeven, is het gebeurtenisobject dat later ook aan de listenermethoden wordt doorgegeven. In dit geval is het een instantie van de klasse AlarmEvent, een subklasse van de klasse Event die speciaal voor dit voorbeeld is gemaakt.

Een aangepaste alarmgebeurtenis doorgeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Alle gebeurtenislisteners ontvangen een gebeurtenisobjectparameter met informatie over de specifieke gebeurtenis die plaatsvindt. In veel gevallen is het gebeurtenisobject een instantie van de klasse Event. In sommige gevallen is het echter handig om aanvullende informatie aan gebeurtenislisteners door te geven. Een algemene manier om dit te bereiken, is om een nieuwe klasse, subklasse of gebeurtenisklasse te definiëren en een instantie van de klasse als het gebeurtenis gebruiken. In dit voorbeeld wordt een instantie AlarmEvent als gebeurtenisobject gebruikt wanneer de gebeurtenis `alarm` van de klasse AlarmClock wordt verzonden. De klasse AlarmEvent, die hier wordt weergegeven, bevat aanvullende informatie over de gebeurtenis `alarm`, met name het wekbericht:

```
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;
    }
    ...
}
```

De beste manier om een aangepaste gebeurtenisobjectklasse te maken, is het definiëren van een klasse die de klasse Event uitbreidt, zoals weergegeven in het vorige voorbeeld. Voor het aanvullen van de overgeërfde functionaliteit definieert de klasse AlarmEvent de eigenschap `message`, die de tekst bevat van het alarmbericht dat aan de gebeurtenis is gekoppeld. De waarde van `message` wordt als parameter doorgegeven in de constructor van AlarmEvent. De klasse AlarmEvent definieert ook de constante `ALARM`, die naar de specifieke gebeurtenis (`alarm`) kan verwijzen bij het aanroepen van de methode `addEventListener()` van de klasse AlarmClock.

Naast het toevoegen van aangepaste functionaliteit, moet elke subklasse van de klasse Event de overgeërfde methode `clone()` overschrijven als onderdeel van het systeem voor gebeurtenisafhandeling van ActionScript. Subklassen van de klasse Event kunnen optioneel ook de overgeërfde methode `toString()` overschrijven om de eigenschappen van de aangepaste gebeurtenis op te nemen in de waarde die bij het aanroepen van de methode `toString()` wordt geretourneerd.

```
    /**
     * 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");
    }
```

De overschreven methode `clone()` moet een nieuwe instantie van de aangepaste Event-subklasse retourneren, waarbij alle aangepaste eigenschappen zo worden ingesteld dat deze met de huidige instantie overeenkomen. In de overschreven methode `toString()` wordt de hulpprogrammamethode `formatToString()` (overgeërfd van Event) gebruikt om een tekenreeks te verkrijgen met de naam van het aangepaste type en met de namen en waarden van alle eigenschappen hiervan.

Hoofdstuk 9: Werken met toepassingsdomeinen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het doel van de klasse ApplicationDomain is het opslaan van een tabel met ActionScript 3.0-definities. Alle code in een SWF-bestand wordt zo gedefinieerd dat deze geschikt is voor een toepassingsdomein. Toepassingsdomeinen worden gebruikt om klassen in te delen die deel uitmaken van hetzelfde beveiligingsdomein. Zo kunnen meerdere definities van dezelfde klasse bestaan en kunnen onderliggende toepassingen de definities van bovenliggende toepassingen hergebruiken.

U kunt toepassingsdomeinen gebruiken wanneer u een extern SWF-bestand dat is geschreven in ActionScript 3.0 laadt met de API van de klasse Loader. (Overigens kunt u geen toepassingsdomeinen gebruiken wanneer u afbeeldingen of SWF-bestanden laadt die zijn geschreven in ActionScript 1.0 of ActionScript 2.0.) Alle ActionScript 3.0-definities die zich in de geladen klasse bevinden, worden opgeslagen in het toepassingsdomein. Wanneer u het SWF-bestand laadt, kunt u opgeven dat het bestand moet worden opgenomen in hetzelfde toepassingsdomein als dat van het object Loader, door de parameter `applicationDomain` van het object LoaderContext in te stellen op `ApplicationDomain.currentDomain`. Als u het geladen SWF-bestand in hetzelfde toepassingsdomein plaatst, hebt u direct toegang tot de klassen van het bestand. Dit kan handig zijn als u een SWF-bestand laadt dat ingesloten media bevat, die u kunt benaderen via de bijbehorende klassennamen, of u als u toegang wilt hebben tot de methoden van het geladen SWF-bestand.

In het volgende voorbeeld wordt ervan uitgegaan dat er toegang is tot een afzonderlijk Greeter.swf-bestand waarmee een openbare methode wordt gedefinieerd met de naam welcome():

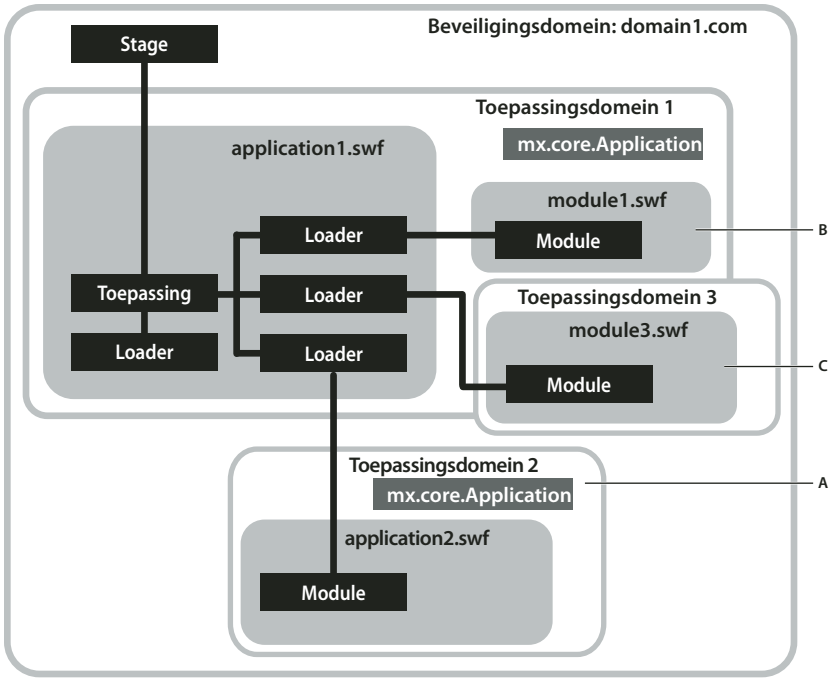
```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLRequest;
    import flash.system.ApplicationDomain;
    import flash.system.LoaderContext;

    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
        }
    }
}
```

Zie ook het voorbeeld van de ApplicationDomain-klasse in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform.

Wanneer u met toepassingsdomeinen werkt, moet u rekening houden met het volgende:

- Alle code in een SWF-bestand wordt zo gedefinieerd dat deze geschikt is voor een toepassingsdomein. Het *huidige domein* is het domein waarin uw hoofdtoepassing wordt uitgevoerd. Het *systeemdomein* bevat alle toepassingsdomeinen, inclusief het huidige domein. Dit betekent dat het alle Flash Player-klassen bevat.
- Aan alle toepassingsdomeinen, met uitzondering van het systeemdomein, is een bovenliggend domein gekoppeld. Het bovenliggende domein van het toepassingsdomein van de hoofdtoepassing is het systeemdomein. Geladen klassen worden alleen gedefinieerd wanneer dit niet wordt gedaan door het bovenliggende domein. U kunt een definitie van een geladen klasse niet overschrijven met een nieuwe definitie.

Het volgende diagram toont een toepassing die inhoud laadt van verschillende SWF-bestanden in één domein, domain1.com. Afhankelijk van de inhoud die u laadt, kunnen verschillende toepassingsdomeinen worden gebruikt. De onderstaande tekst beschrijft de logica die wordt gebruikt om het juiste toepassingsdomein in te stellen voor elk SWF-bestand in de toepassing.



***A.** Gebruik A: **B.** Gebruik B **C.** Gebruik C*

Het hoofdtoepassingsbestand is application1.swf. Het bevat objecten Loader die inhoud laden van andere SWF-bestanden. In dit scenario is het huidige domein toepassingsdomein 1. Gebruik A, B en C tonen verschillende technieken voor het instellen van het juiste toepassingsdomein voor elk SWF-bestand in een toepassing.

Gebruik A: het onderliggende SWF-bestand afscheiden door een onderliggend domein van het systeemdomein te maken. In het diagram wordt toepassingsdomein 2 gemaakt als een onderliggend domein van het systeemdomen. Het bestand application2.swf wordt geladen in toepassingsdomein 2, zodat de klassendefinities worden afgescheiden van de klassen die zijn gedefinieerd in application1.swf.

U kunt deze techniek bijvoorbeeld gebruiken om een oude toepassing een nieuwe versie van dezelfde toepassing te laten laden zonder dat er een conflict optreedt. Hoewel dezelfde klassennamen worden gebruikt, ontstaat er geen conflict omdat deze worden ondergebracht in verschillende toepassingsdomeinen.

Met de volgende code wordt een toepassingsdomein gemaakt dat een onderliggend domein van het systeemdomein is. Vervolgens wordt met dat toepassingsdomein een SWF geladen.

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

Gebruik B: nieuwe klassendefinities toevoegen aan de huidige klassendefinities. Het toepassingsdomein van module1.swf wordt ingesteld op het huidige domein (toepassingsdomein 1). U kunt nu nieuwe klassendefinities toevoegen aan de huidige set klassendefinities van de toepassing. U kunt dit bijvoorbeeld doen voor een gezamenlijke bibliotheek bij uitvoering van de hoofdtoepassing. De geladen SWF wordt behandeld als een externe gezamenlijke bibliotheek (RSL). U kunt deze techniek gebruiken om RSL's te laden via een voorlader voordat de toepassing wordt gestart.

Met de volgende code wordt een SWF geladen, waarbij het toepassingsdomein wordt ingesteld op het huidige domein:

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

Gebruik C: de klassendefinities van de bovenliggende toepassing gebruiken door een nieuw onderliggend domein aan te maken van het huidige domein. Het toepassingsdomein van module3.swf is een onderliggend domein van het huidige domein en de onderliggende toepassing gebruikt de versies van de bovenliggende toepassing van alle klassen. U kunt deze techniek bijvoorbeeld gebruiken om een module van een RIA (Rich Internet Application) met meerdere schermen te laden als een onderliggende toepassing van de hoofdtoepassing, die de typen van de hoofdtoepassing gebruikt. Als u ervoor kunt zorgen dat alle klassen altijd achterwaarts compatibel blijven als ze worden bijgewerkt, en dat de ladende toepassing altijd nieuwer is dan wat er wordt geladen, gebruiken de onderliggende toepassingen altijd de versies van de bovenliggende toepassingen. Met een nieuw toepassingsdomein kunt u bovendien alle klassendefinities verwijderen, zolang u er maar voor zorgt dat u alle referenties naar de onderliggende SWF ook verwijdert.

Met deze techniek kunnen geladen modules de objecten Singleton en statische klassenleden delen.

Met de volgende code wordt een nieuw onderliggend domein van het huidige domein gemaakt. Vervolgens wordt met dat toepassingsdomein een SWF geladen.

```
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);
```

Hoofdstuk 10: Weergave programmeren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

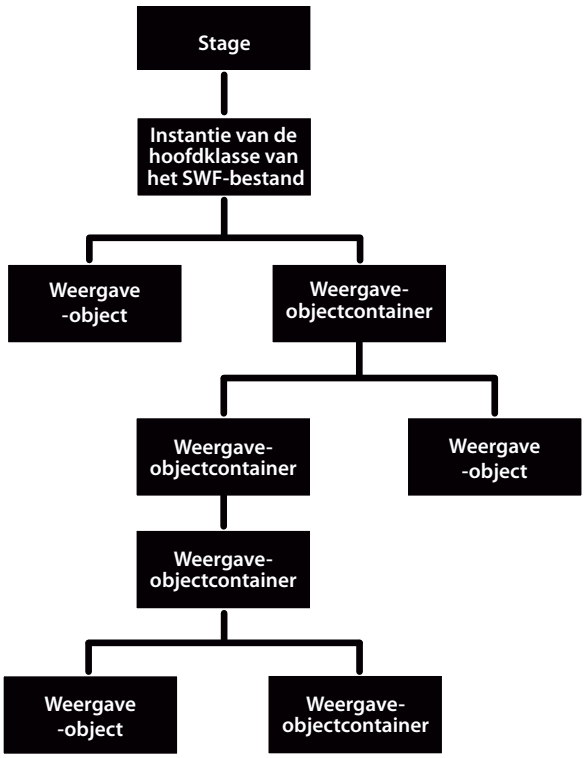
Visuele elementen worden geprogrammeerd in Adobe® ActionScript® 3.0 met behulp van weergaveobjecten in het weergavegebied. U kunt bijvoorbeeld weergaveobjecten toevoegen, verplaatsen, verwijderen, ordenen en weergeven, filters en maskers toepassen, vector- en bitmapafbeeldingen en driedimensionale transformaties uitvoeren met behulp van de API voor het programmeren van de ActionScript-weergave. De primaire klassen die worden gebruikt voor weergaveprogrammering zijn onderdeel van het flash.display-pakket.

Opmerking: *Adobe® AIR™ biedt het HTMLoader-object voor het renderen en weergeven van HTML-inhoud. De HTMLLoader rendert de visuele elementen van de HTML DOM als één weergaveobject. U hebt geen rechtstreekse toegang tot individuele elementen van de DOM via de hiërarchie in de weergavelijst van ActionScript. U kunt deze DOM-elementen in plaats daarvan openen met de aparte DOM API die bij de HTMLLoader is meegeleverd.*

Basisbeginselen van weergave programmeren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elke toepassing die met ActionScript 3.0 wordt gemaakt, heeft een hiërarchie van weergegeven objecten, de zogenaamde *weergavelijst* (zie de volgende illustratie). De weergavelijst bevat alle zichtbare elementen in de toepassing.



Zoals u in de illustratie ziet, behoren weergave-elementen tot een of meer van de volgende groepen:

- Het werkgebied

  Het werkgebied is de basiscontainer voor weergaveobjecten. Elke toepassing heeft een object Stage, dat alle weergaveobjecten op het scherm bevat. Het werkgebied is de container op hoofdniveau en bevindt zich helemaal bovenaan in de hiërarchie van het weergaveoverzicht:

  Elk SWF-bestand heeft een gekoppelde ActionScript-klasse, die *de hoofdklasse van het SWF-bestand* is. Wanneer een SWF-bestand wordt geopend in Flash Player of Adobe AIR, roept Flash Player of AIR de constructorfunctie voor die klasse aan. De instantie die wordt gemaakt (altijd een type weergaveobject) wordt toegevoegd als een onderliggende instantie van het object Stage. De hoofdklasse van een SWF-bestand is altijd een uitbreiding van de klasse Sprite (zie "Voordelen van de weergaveoverzichtaanpak" op pagina 166).

  U hebt toegang tot het werkgebied (Stage) door de eigenschap `stage` van een instantie DisplayObject te gebruiken. Zie "De eigenschappen van Stage instellen" op pagina 175 voor meer informatie.

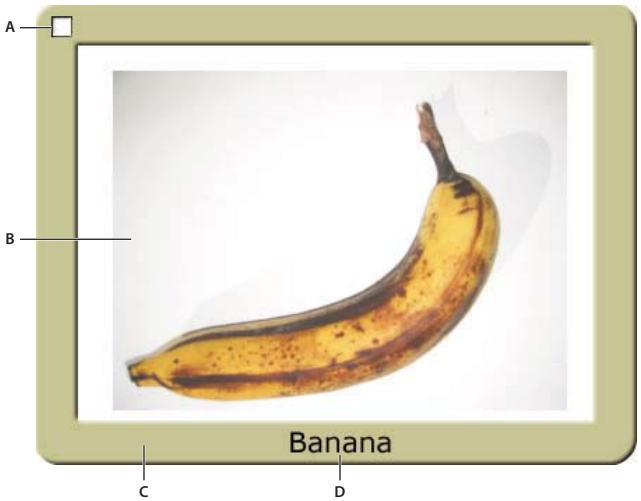
- Weergaveobjecten

  In ActionScript 3.0 zijn alle elementen die in een toepassing op het scherm worden getoond, typen *weergaveobjecten*. Het pakket flash.display bevat de klasse DisplayObject . Dit is een basisklasse die met een aantal andere klassen is uitgebreid. Deze verschillende klassen vertegenwoordigen de verschillende typen weergaveobjecten, zoals vectorvormen, filmclips en tekstvelden. Zie "Voordelen van de weergaveoverzichtaanpak" op pagina 166 voor een overzicht van deze klassen.

- Weergaveobjectcontainers

  Weergaveobjectcontainers zijn speciale typen weergaveobjecten die, naast hun eigen visuele representatie, onderliggende objecten kunnen bevatten die ook weergaveobjecten zijn.

  De klasse DisplayObjectContainer is een subklasse van de klasse DisplayObject. Een object DisplayObjectContainer kan meerdere weergaveobjecten inde bijbehorende *lijst met onderliggende objecten* bevatten. De onderstaande afbeelding toont bijvoorbeeld een objecttype DisplayObjectContainer, bekend als Sprite, dat verschillende weergaveobjecten bevat:



***A.** Een object SimpleButton. Dit type weergaveobject heeft verschillende toestanden: omhoog, omlaag en boven. **B.** Een object Bitmap. In dit geval is het object Bitmap via een object Loader vanuit een externe JPEG geladen. **C.** Een object Shape. De schilderijlijst bevat een afgeronde rechthoek die in ActionScript is getekend. Er is een filter Slagschaduw toegepast op dit object Shape. **D.** Een object TextField.*

De DisplayObjectContainer-objecten worden ook *weergaveobjectcontainers* of *containers* genoemd. Zoals eerder opgemerkt, is Stage een weergaveobjectcontainer.

Hoewel alle zichtbare weergaveobjecten van de klasse DisplayObject overerven, is het type van elk een specifieke subklasse van de klasse DisplayObject. Er bestaat bijvoorbeeld een constructorfunctie voor de klasse Shape en de klasse Video, maar er is geen constructorfunctie voor de klasse DisplayObject.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het programmeren van ActionScript-afbeeldingen:

Alfa De kleurwaarde die de hoeveelheid transparantie (of beter, de hoeveelheid dekking) in een kleur vertegenwoordigt. Een kleur met een alfakanaalwaarde van bijvoorbeeld 60% laat slechts 60% van zijn volledige sterkte zien en is 40% transparant.

Bitmapafbeelding Een afbeelding die in de computer wordt gedefinieerd als een raster (rijen en kolommen) van gekleurde pixels. Digitale foto's en vergelijkbare afbeeldingen zijn doorgaans bitmapafbeeldingen.

Overvloeimodus Een specificatie van hoe de inhoud van twee overlappende afbeeldingen zou moeten communiceren. Een dekkende afbeelding boven een andere afbeelding schermt gewoonlijk de onderliggende afbeelding af, zodat deze niet zichtbaar is. Verschillende overvloeimodi zorgen er echter voor dat de afbeeldingen op verschillende manieren samenvloeien, zodat de resulterende inhoud een combinatie van de twee afbeeldingen vormt.

Weergaveoverzicht De hiërarchie van weergaveobjecten die door Flash Player en AIR worden gerenderd als zichtbare inhoud op het scherm. Het werkgebied is de basis van het weergaveoverzicht en alle weergaveobjecten die aan het werkgebied of één van de onderliggende objecten zijn gekoppeld, vormen het weergaveoverzicht (zelfs als het object niet wordt gerenderd, bijvoorbeeld als het zich buiten de grenzen van het werkgebied bevindt).

Weergaveobject Een object dat een type visuele inhoud vertegenwoordigt in Flash Player of AIR. Alleen weergaveobjecten kunnen worden opgenomen in het weergaveoverzicht en alle weergaveobjectklassen zijn subklassen van de klasse DisplayObject.

Weergaveobjectcontainer Een speciaal type weergaveobject dat onderliggende weergaveobjecten kan bevatten naast (gewoonlijk) zijn eigen visuele representatie.

Hoofdklasse van het SWF-bestand De klasse die het gedrag van het buitenste weergaveobject in een SWF-bestand definieert. Dit is de klasse voor het SWF-bestand zelf. In een SWF-bestand dat in de Flash-ontwerpomgeving is gemaakt, is de hoofdklasse bijvoorbeeld de documentklasse. Het heeft een hoofdtijdlijn, die alle andere tijdlijnen bevat. De hoofdklasse van het SWF-bestand is de klasse waarvan de hoofdtijdlijn een instantie is.

Maskeren Een techniek waarmee bepaalde delen van een afbeelding worden verborgen (of omgekeerd, waarmee alleen bepaalde delen van een afbeelding worden weergegeven). De delen van de gemaskeerde afbeelding worden transparant, zodat de onderliggende inhoud zichtbaar wordt. De term houdt verband met afplakband (masking tape) van een schilder, dat wordt gebruikt om te voorkomen dat verf op bepaalde gedeelten wordt gemorst.

Werkgebied De visuele container die de basis of achtergrond voor alle visuele inhoud in een SWF vormt.

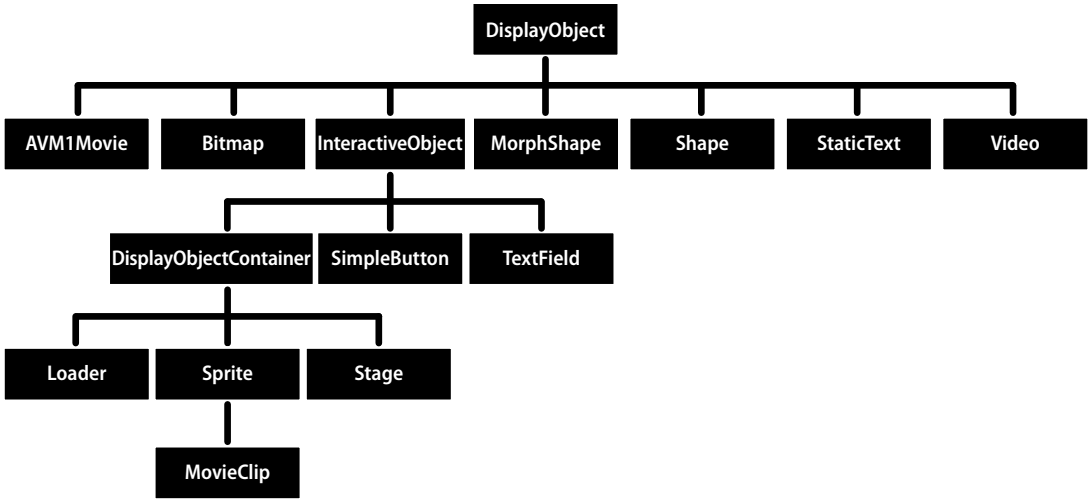
Transformatie Een aanpassing in een visueel kenmerk van een afbeelding, zoals een object roteren, schalen, scheeftrekken of vervormen of de kleur ervan wijzigen.

Vectorafbeelding Een afbeelding die in de computer wordt gedefinieerd als lijnen en vormen die met bepaalde kenmerken worden getekend (bijvoorbeeld dikte, lengte, grootte, hoek en positie).

Kernweergaveklassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het pakket flash.display van ActionScript 3.0 bevat klassen voor visuele objecten die in Flash Player of AIR kunnen worden weergegeven. De volgende afbeelding toont de verhoudingen tussen de subklassen van deze kernweergaveklassen.



De afbeelding toont de klassenovererving van weergaveobjectklassen. Enkele van deze klassen, vooral StaticText, TextField en Video, zitten niet in het pakket flash.display maar overerven wel van de klasse DisplayObject.

Alle klassen die de klasse DisplayObject uitbreiden, overerven de methoden en eigenschappen hiervan. Zie "Eigenschappen en methoden van de klasse DisplayObject" op pagina 169 voor meer informatie.

U kunt objecten van de volgende klassen instantiëren die in het pakket flash.display zitten:

- Bitmap - U gebruikt de klasse Bitmap om bitmapobjecten te definiëren die vanuit externe bestanden zijn geladen of via ActionScript zijn gerenderd. Met de klasse Loader kunt u bitmaps vanuit externe bestanden laden. U kunt GIF-, JPG- of PNG-bestanden laden. U kunt ook een object BitmapData met aangepaste gegevens maken, waarna u een object Bitmap maakt dat deze gegevens gebruikt. U kunt de methoden van de klasse BitmapData gebruiken om bitmaps te wijzigen, ongeacht of ze in ActionScript zijn gemaakt of worden geladen. Zie "Weergaveobjecten laden" op pagina 211 en "Werken met bitmaps" op pagina 256 voor meer informatie.
- Loader - U kunt de klasse Loader gebruiken om externe elementen te laden (SWF-bestanden of afbeeldingen). Zie "Scherminhoud dynamisch laden" op pagina 210 voor meer informatie.
- Shape - U kunt de klasse Shape gebruiken om vectorafbeeldingen te maken, zoals rechthoeken, lijnen, cirkels enzovoort. Zie "De teken-API gebruiken" op pagina 235 voor meer informatie.
- SimpleButton - een SimpleButton-object is de ActionScript-representatie van een knopsymbool dat met het Flash-programma voor het schrijven van programmacode is gemaakt. Een instantie SimpleButton heeft vier knoptoestanden: omhoog, omlaag, boven en detectietest (het gebied dat reageert op muis- en toetsenbordgebeurtenissen).
- Sprite - Een object Sprite kan zelf afbeeldingen en onderliggende weergaveobjecten bevatten. (De klasse Sprite breidt de klasse DisplayObjectContainer uit.) Zie "Werken met weergaveobjectcontainers" op pagina 169 en "De teken-API gebruiken" op pagina 235 voor meer informatie.

- MovieClip - een MovieClip-object is de ActionScript-vorm van een filmfragmentsymbool dat met het Flash-programma voor het schrijven van programmacode is gemaakt. In de praktijk is een object MovieClip vergelijkbaar met een object Sprite, behalve dat deze ook een tijdlijn heeft. Zie "Werken met filmclips" op pagina 341 voor meer informatie.

De volgende klassen, die niet in het pakket flash.display zitten, zijn subklassen van de klasse DisplayObject:

- De klasse TextField, die in het pakket flash.text zit, is een weergaveobject voor tekstweergave en tekstinvoer. Zie "Basisbeginselen van het werken met tekst" op pagina 391 voor meer informatie.
- De klasse TextLine, die in het pakket flash.text.engine zit, is het weergaveobject waarmee tekstregels worden weergegeven die zijn samengesteld met de Flash Text Engine en Text Layout Framework. Zie "De Flash Text Engine gebruiken" op pagina 418 en "Text Layout Framework gebruiken" op pagina 448 voor meer informatie.
- De klasse Video, die in het pakket flash.media zit, is het weergaveobject dat wordt gebruikt voor het weergeven van videobestanden. Zie "Werken met video" op pagina 499 voor meer informatie.

De volgende klassen in het pakket flash.display breiden de klasse DisplayObject uit, maar er kunnen geen instanties van worden gemaakt. In plaats hiervan dienen ze als hoofdklassen voor overige weergaveobjecten waarbij algemene functies tot een enkele klasse worden gecombineerd.

- AVM1Movie - De klasse AVM1Movie wordt gebruikt om geladen SWF-bestanden te vertegenwoordigen die zijn bewerkt in ActionScript 1.0 en 2.0.
- DisplayObjectContainer - De klassen Loader, Stage, Sprite en MovieClip breiden allemaal de klasse DisplayObjectContainer uit. Zie "Werken met weergaveobjectcontainers" op pagina 169 voor meer informatie.
- InteractiveObject - InteractiveObject is de basisklasse voor alle objecten die worden gebruikt voor interactie met de muis en het toetsenbord. Objecten SimpleButton, TextField, Loader, Sprite, Stage en MovieClip zijn allemaal subklassen van de klasse InteractiveObject. Zie "Basis van gebruikersinteractie" op pagina 586 voor meer informatie over het maken van interactie met de muis en het toetsenbord.
- MorphShape - Deze objecten worden gemaakt wanneer u met het Flash-ontwerpgereedschap een vormtween maakt. U kunt ze met ActionScript niet instantiëren, maar u hebt wel toegang tot deze objecten via het weergaveoverzicht.
- Stage - De klasse Stage breidt de klasse DisplayObjectContainer uit. Er bestaat één werkgebiedinstantie voor een toepassing en deze bevindt zich op het hoogste niveau van de hiërarchie van het weergaveoverzicht. Als u toegang wilt tot het werkgebied, gebruikt u de eigenschap `stage` van een instantie DisplayObject. Zie "De eigenschappen van Stage instellen" op pagina 175 voor meer informatie.

De klasse StaticText, in het pakket flash.text, breidt de klasse DisplayObject ook uit, maar u kunt hier geen instantie van in de code maken. Statische tekstvelden worden alleen in Flash gemaakt.

De volgende klassen zijn geen weergaveobjecten of weergaveobjectcontainers en komen niet voor in de weergavelijst, maar geven wel afbeeldingen weer in het werkgebied. Deze klassen tekenen een rechthoek, een zogenaamde viewport, die ten opzichte van het werkgebied wordt geplaatst.

- StageVideo—De klasse StageVideo geeft video-inhoud weer, waar mogelijk met hardwareversnelling. Deze klasse is beschikbaar vanaf Flash Player 10.2. Zie "De klasse StageVideo gebruiken voor presentatie met hardwareversnelling" op pagina 539 voor meer informatie.
- StageWebView—De klasse StageWebView geeft HTML-inhoud weer. Deze klasse is beschikbaar vanaf AIR 2.5. Zie "StageWebView-objecten" op pagina 1089 voor meer informatie.

De volgende fl.display-klassen bieden functionaliteit die overeenkomt met de klassen flash.display.Loader en LoaderInfo. Gebruik deze klassen in plaats van de flash.display-equivalenten als u in de Flash Professional-omgeving (CS5.5 of hoger) ontwikkelt. In die omgeving kunt u met deze klassen problemen oplossen met betrekking tot het voorladen van TLF met RSL. Zie "De klassen ProLoader en ProLoaderInfo gebruiken" op pagina 215 voor meer informatie.

- fl.display.ProLoader—Analoog aan flash.display.Loader
- fl.display.ProLoaderInfo—Analoog aan flash.display.LoaderInfo

Voordelen van de weergaveoverzichtaanpak

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 3.0 bestaan aparte klassen voor verschillende type weergaveobjecten. In ActionScript 1.0 en 2.0 bevinden veel van dezelfde type objecten zich in één klasse: de klasse MovieClip.

Deze klassenscheiding en de hiërarchische structuur van het weergaveoverzicht hebben de volgende voordelen:

- Verbeterde renderingprestaties en minder geheugengebruik
- Verbeterd dieptebeheer
- Het volledig doorlopen van het weergaveoverzicht
- Niet-weergegeven weergaveobjecten
- Vereenvoudigde subclassificering van weergaveobjecten

Verbeterde renderingprestaties en kleinere bestandsgrootten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 1.0 en 2.0 kunt u vormen alleen in een object MovieClip tekenen. In ActionScript 3.0 hebt u eenvoudigere weergaveobjectklassen tot uw beschikking waarin u vormen kunt tekenen. Omdat de weergaveobjectklassen van ActionScript 3.0 niet alle methoden en eigenschappen hebben die een object MovieClip heeft, nemen ze minder geheugen en processorresources in beslag.

Elk object MovieClip bevat bijvoorbeeld eigenschappen voor de tijdlijn van een filmclip en een object Shape niet. De eigenschappen voor het beheren van de tijdlijn nemen veel geheugen en processorresources in beslag. In ActionScript 3.0 behaalt u betere prestaties als u het object Shape gebruikt. Het object Shape heeft minder overhead dan het meer complexe object MovieClip. Flash Player en AIR hoeven de eigenschappen MovieClip die niet worden gebruikt ook niet te beheren, waardoor de snelheid wordt verhoogd en het geheugengebruik wordt verminderd.

Verbeterd dieptebeheer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 1.0 en 2.0 wordt diepte beheerd via een lineair dieptebeheerschema en methoden zoals `getNextHighestDepth()`.

ActionScript 3.0 bevat de klasse DisplayObjectContainer die betere methoden en eigenschappen voor het beheer van diepteobjecten en weergaveobjecten biedt.

Wanneer u in ActionScript 3.0 een weergaveobject naar een nieuwe positie in de lijst met onderliggende items van een instantie DisplayObjectContainer verplaatst, wordt de positie van de overige onderliggende inhoud van de weergaveobjectcontainer automatisch gewijzigd en krijgen ze automatisch de juiste onderliggende indexposities in de weergaveobjectcontainer toegewezen.

In ActionScript 3.0 is het ook altijd mogelijk om alle onderliggende objecten van een weergaveobjectcontainer te vinden. Elke instantie DisplayObjectContainer bevat de eigenschap `numChildren`, die het aantal onderliggende objecten in de weergaveobjectcontainer opsomt. Omdat de lijst met onderliggende items van een weergaveobjectcontainer altijd een geïndexeerd overzicht is, kunt u elk object in het overzicht, vanaf indexpositie 0 tot de laatste indexpositie, laten weergeven (`numChildren - 1`). Dit was niet mogelijk met de methoden en eigenschappen van een object MovieClip in ActionScript 1.0 en 2.0.

In ActionScript 3.0 kunt u het weergaveoverzicht eenvoudig opeenvolgend doorlopen, er bevinden zich geen tussenruimten tussen de indexnummers van een lijst met onderliggende items van een weergaveobjectcontainer. Het doorlopen van het weergaveoverzicht en het beheren van de diepte van objecten is nu veel eenvoudiger dan in ActionScript 1.0 en 2.0 het geval was. In ActionScript 1.0 en 2.0 kon een filmclip objecten met periodieke tussenruimten in de dieptevolgorde bevatten, wat het moeilijk maakt om het overzicht van objecten te doorlopen. In ActionScript 3.0 wordt elke lijst met onderliggende items van een weergaveobjectcontainer intern in cache geplaatst, waardoor er zeer snel kan worden gezocht (op index). Het doorlopen van de overige onderliggende inhoud van een weergaveobjectcontainer gebeurt ook zeer snel.

In ActionScript 3.0 hebt u alleen toegang tot de onderliggende inhoud van een weergaveobjectcontainer via de methode `getChildByName()` van de klasse DisplayObjectContainer.

Het volledig doorlopen van het weergaveoverzicht

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 1.0 en 2.0 hebt u soms geen toegang tot sommige objecten, zoals vectorvormen, die zijn getekend met het Flash-ontwerpgereedschap. In ActionScript 3.0 hebt u toegang tot alle objecten in het weergaveoverzicht, zowel de objecten die zijn gemaakt met ActionScript en alle weergaveobjecten die zijn gemaakt met het Flash-ontwerpgereedschap. Zie "Het weergaveoverzicht doorlopen" op pagina 173 voor meer informatie.

Niet-weergegeven weergaveobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 3.0 kunt u weergaveobjecten maken die niet op het zichtbare weergaveoverzicht worden geplaatst. Dit worden *niet-weergegeven* weergaveobjecten genoemd. Een weergaveobject wordt alleen aan het zichtbare weergaveoverzicht toegevoegd wanneer u de methode `addChild()` of `addChildAt()` van een instantie DisplayObjectContainer aanroept, die reeds aan het weergaveoverzicht is toegevoegd.

U kunt de niet-weergegeven weergaveobjecten gebruiken om complexe weergaveobjecten samen te stellen, zoals weergaveobjecten die meerdere weergaveobjectcontainers met meerdere weergaveobjecten bevatten. Door weergaveobjecten niet weer te geven, kunt u complexe objecten samenstellen zonder verwerkingstijd in beslag te nemen voor het renderen van deze weergaveobjecten. Wanneer gewenst kunt u een niet-weergegeven weergaveobject aan het weergaveoverzicht toevoegen. Ook kunt u een onderliggend item in de weergaveobjectcontainer in het weergaveoverzicht plaatsen of deze van de lijst verwijderen en u kunt de items naar elke gewenste positie op de lijst verplaatsen.

Vereenvoudigde subclassificering van weergaveobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 1.0 en 2.0 moet u vaak nieuwe objecten MovieClip aan een SWF-bestand toevoegen om standaardvormen te maken of om bitmaps weer te geven. In ActionScript 3.0 bevat de klasse DisplayObject veel ingebouwde subklassen, waaronder Shape en Bitmap. Omdat de klassen in ActionScript 3.0 gespecialiseerd zijn voor bepaalde typen objecten, is het eenvoudiger om basissubklassen van de ingebouwde klassen te maken.

Als u bijvoorbeeld een cirkel in ActionScript 2.0 wilt tekenen, kunt u een klasse CustomCircle maken die de klasse MovieClip uitbreidt wanneer een object van de aangepaste klasse wordt geïnstantieerd. Die klasse zou echter ook een aantal eigenschappen en methoden van de klasse MovieClip bevatten (zoals `totalFrames`) die niet van toepassing op de klasse zijn. Maar in ActionScript 3.0 kunt u een klasse CustomCircle maken die het object Shape uitbreidt en die niet de irrelevante eigenschappen en methoden bevat die zich in de klasse MovieClip bevinden. De volgende code toont een voorbeeld van een klasse 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);
    }
}
```

Werken met weergaveobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Nu dat u de basisconcepten van het werkgebied, weergaveobjecten, weergaveobjectcontainers en het weergaveoverzicht begrijpt, wordt in deze sectie dieper ingegaan op het werken met weergaveobjecten in ActionScript 3.0.

Eigenschappen en methoden van de klasse DisplayObject

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Alle weergaveobjecten zijn subklassen van de klasse DisplayObject en overerven daarom de eigenschappen en methoden van de klasse DisplayObject. De overgeërfde eigenschappen zijn basiseigenschappen die op alle weergaveobjecten van toepassing zijn. Elk weergaveobject heeft bijvoorbeeld een eigenschap `x` en een eigenschap `y` die de positie van het object in zijn weergaveobjectcontainer opgeven.

U kunt geen instantie DisplayObject met de klassenconstructor DisplayObject maken. U moet een ander type object maken (een object dat een subklasse van de klasse DisplayObject is), zoals een Sprite, om een object zonder de operator `new` te instantiëren. Als u een aangepaste weergaveobjectklasse wilt maken, moet u een subklasse van een van de weergaveobjectsubklassen maken die een bruikbare constructorfunctie heeft (zoals een Shape of Sprite). Zie de beschrijving van de DisplayObject-klasse in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Weergaveobjecten aan het weergaveoverzicht toevoegen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een weergaveobject instantieert, wordt het object niet (in het werkgebied) op het scherm weergegeven tot u de weergaveobjectinstantie aan een weergaveobjectcontainer toevoegt, die zich in het weergaveoverzicht bevindt. In de volgende code is het TextField-object `myText` niet zichtbaar als u de laatste regel code vergeet. In de laatste coderegel moet het trefwoord `this` verwijzen naar een weergaveobjectcontainer die al aan het weergaveoverzicht is toegevoegd.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

Als u een visueel element aan het werkgebied toevoegt, wordt dat element een *onderliggend element* van het object Stage. Het eerste SWF-bestand dat in een toepassing wordt geladen (bijvoorbeeld het bestand dat u in een HTML-pagina insluit) wordt automatisch toegevoegd als een onderliggend bestand van het werkgebied. Het kan een object van een willekeurig type zijn dat de klasse Sprite uitbreidt.

De weergaveobjecten die u *zonder* ActionScript maakt (door bijvoorbeeld een MXML-tag in een Flex MXML-bestand toe te voegen of door een item op het werkgebied in Flash te plaatsen) worden aan de weergavelijst toegevoegd. Hoewel u deze objecten niet met ActionScript toevoegt, hebt u wel toegang tot deze objecten via ActionScript. De volgende code past bijvoorbeeld de breedte van een object met de naam `button1` aan, dat met het ontwerpgereedschap is toegevoegd (niet met ActionScript):

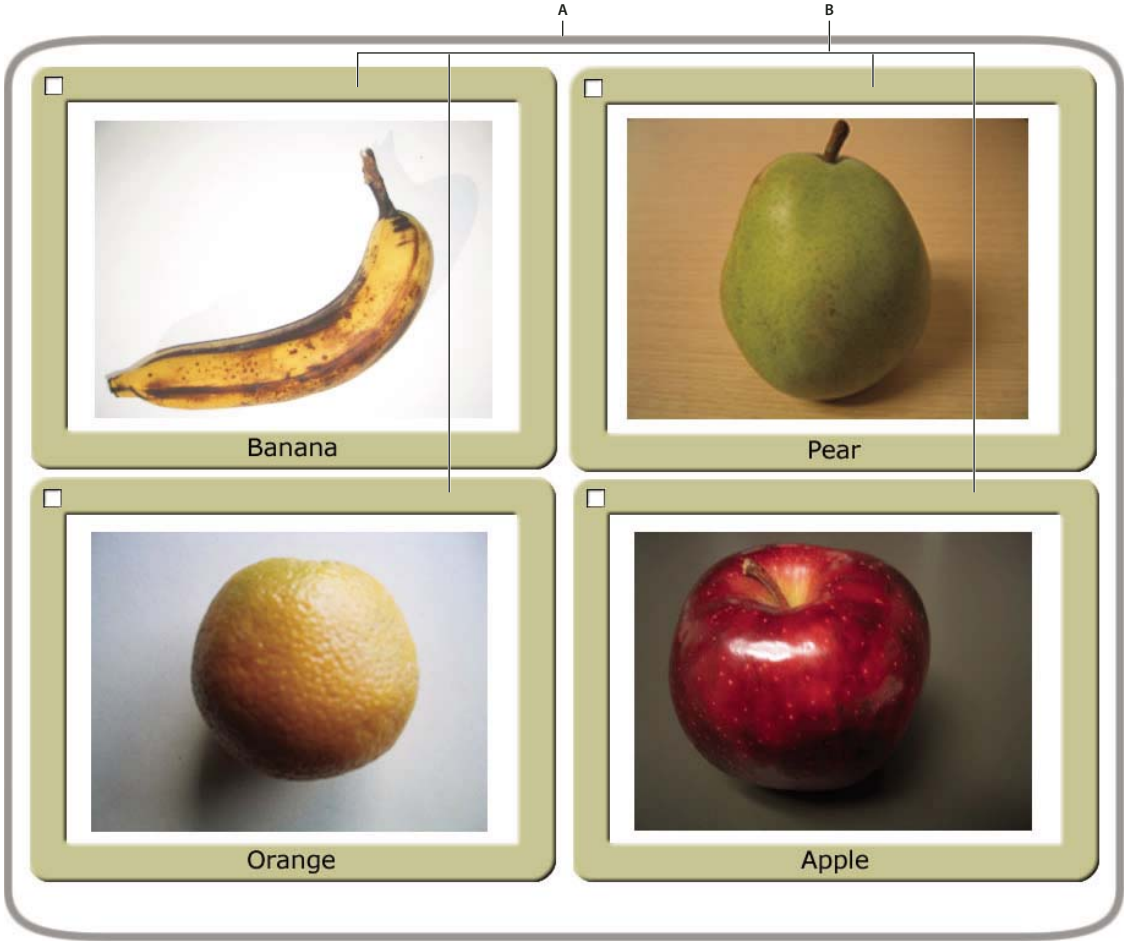
```
button1.width = 200;
```

Werken met weergaveobjectcontainers

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als een object DisplayObjectContainer uit het weergaveoverzicht wordt verwijderd, wordt verplaatst of wordt getransformeerd, wordt elk weergaveobject in de DisplayObjectContainer ook verwijderd, verplaatst of getransformeerd.

Een weergaveobjectcontainer is zelf een type weergaveobject; het kan aan een andere weergaveobjectcontainer worden toegevoegd. De volgende afbeelding toont bijvoorbeeld een container voor weergaveobjecten `pictureScreen`, die een lijncontour en vier andere weergaveobjectcontainers bevat (van het type PictureFrame):



***A.** Een vorm waarmee de rand van de weergaveobjectcontainer pictureScreen wordt gedefinieerd **B.** Vier onderliggende weergaveobjectcontainers van het object pictureScreen*

Als u een weergaveobject in het weergaveoverzicht wilt weergeven, moet u deze toevoegen aan de weergaveobjectcontainer die zich in het weergaveoverzicht bevindt. U kunt dit via de methode `addChild()` of de methode `addChildAt()` van het object container doen. Zonder de laatste regel van de volgende code zou het object `myTextField` bijvoorbeeld niet worden weergegeven:

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

In ditzelfde voorbeeld wijst `this.root` naar de weergaveobjectcontainer MovieClip die de code bevat. In uw eigen code moet u een andere container opgeven.

Gebruik de methode `addChildAt()` om het element op een bepaalde positie aan de lijst met onderliggende items van de weergaveobjectcontainer toe te voegen. Deze op nul gebaseerde indexposities in de lijst met onderliggende items hebben betrekking op de laag waarop de weergaveobjecten worden geplaatst (van voor naar achter). Bekijk bijvoorbeeld de volgende drie weergaveobjecten. Elk object is gemaakt op basis van een aangepaste klasse met de naam Ball.

De containerlaag van deze weergaveobjecten kan worden aangepast met de methode `addChildAt()`. Neem bijvoorbeeld de volgende code:

```
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);
```

Nadat u deze code hebt uitgevoerd, worden de weergaveobjecten als volgt geordend in het DisplayObjectContainer-object `container`. Zie de lagen van de objecten.

Als u een object boven aan het weergaveoverzicht wilt plaatsen, voegt u het object opnieuw aan de lijst toe. Als u, op basis van de vorige code, `ball_A` naar de bovenkant van de stapel wilt verplaatsen, gebruikt u de onderstaande coderegel:

```
container.addChild(ball_A);
```

Deze code verwijdert `ball_A` van zijn locatie in het weergaveoverzicht van de `container` en voegt zichzelf opnieuw aan de bovenkant van het overzicht toe. Hiermee wordt het object boven op de stapel geplaatst.

U kunt de methode `getChildAt()` gebruiken om de laagvolgorde van de weergaveobjecten te verifiëren. De methode `getChildAt()` retourneert de onderliggende objecten van een container op basis van het indexnummer dat u doorgeeft. De volgende code onthult bijvoorbeeld de namen van weergaveobjecten op verschillende posities in de lijst met onderliggende items van het DisplayObjectContainer-object `container`:

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

Als u een weergaveobject uit de lijst met onderliggende items van de bovenliggende container verwijdert, worden de daarboven gelegen elementen één positie naar beneden geschoven in de onderliggende index. Op basis van de vorige code, toont de volgende code hoe het weergaveobject dat zich op positie 2 in de `container` DisplayObjectContainer bevindt, naar positie 1 wordt verplaatst als een lager gelegen weergaveobject in de lijst met onderliggende items wordt verwijderd:

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

De methoden `removeChild()` en `removeChildAt()` verwijderen een weergaveobjectinstantie niet geheel. Ze verwijderen het uit de lijst met onderliggende items van de container. Er kan nog steeds door een andere variabele naar de instantie worden verwezen. (Gebruik de operator `delete` om een object volledig te verwijderen.)

Omdat een weergaveobject slechts een bovenliggende container heeft, kunt u een instantie van een weergaveobject slechts aan één weergaveobjectcontainer toevoegen. De volgende code toont bijvoorbeeld dat het weergaveobject `tf1` alleen in een container kan bestaan (in dit geval een Sprite die de klasse DisplayObjectContainer uitbreidt):

```
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
```

Als u een weergaveobject in een weergaveobjectcontainer aan een andere weergaveobjectcontainer toevoegt, wordt deze verwijderd uit de eerste lijst met onderliggende items van de weergaveobjectcontainer.

Naast de hierboven beschreven methoden, definieert de klasse DisplayObjectContainer verschillende methoden voor het werken met onderliggende weergaveobjecten, waaronder:

- `contains()`: hiermee wordt bepaald of een weergaveobject een onderliggend object van een DisplayObjectContainer is.
- `getChildByName()`: hiermee wordt een weergaveobject op naam opgehaald.
- `getChildIndex()`: hiermee wordt de indexpositie van een weergaveobject geretourneerd.
- `setChildIndex()`: hiermee wordt de positie van een onderliggend weergaveobject gewijzigd.
- `removeChildren()`: hiermee worden meerdere onderliggende weergaveobjecten verwijderd.

- `swapChildren()`: hiermee wordt de volgorde (van voor naar achter) van twee weergaveobjecten verwisseld.
- `swapChildrenAt()`: hiermee wordt de volgorde (van voor naar achter) van twee weergaveobjecten verwisseld, opgegeven door hun indexwaarden.

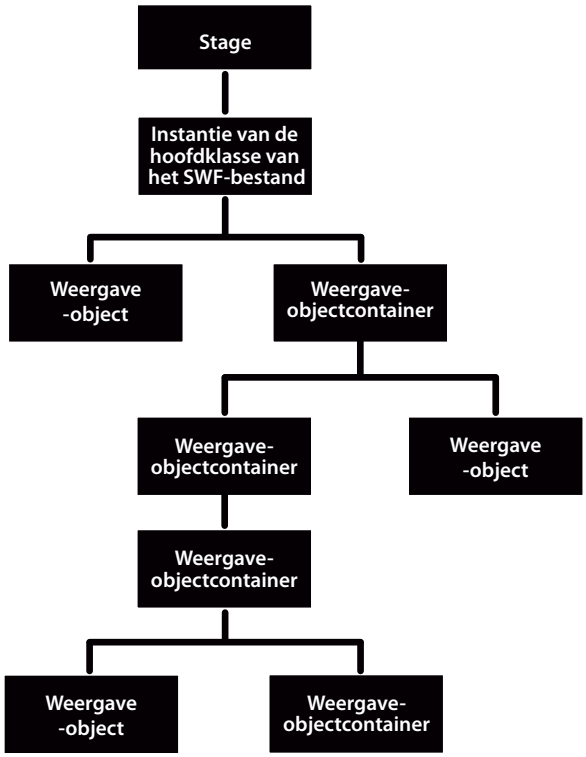
Zie de relevante onderwerpen in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Let erop dat een weergaveobject dat niet in het weergaveoverzicht staat (een object dat niet is opgenomen in een onderliggende weergaveobjectcontainer van het werkgebied) bekend staat als een *niet-weergegeven* weergaveobject.

Het weergaveoverzicht doorlopen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals u weet, heeft het weergaveoverzicht een boomstructuur. Aan de bovenkant van de structuur bevindt zich het werkgebied, dat meerdere weergaveobjecten kan bevatten. Deze weergaveobjecten, die zelf weergaveobjectcontainers zijn, kunnen andere weergaveobjecten of weergaveobjectcontainers bevatten.



De klasse DisplayObjectContainer bevat eigenschappen en methoden voor het doorlopen van het weergaveoverzicht, dit gebeurt via de lijsten met onderliggende items van de weergaveobjectcontainers. Neem bijvoorbeeld de volgende code. Deze code voegt de twee weergaveobjecten `title` en `pict` aan het object `container` toe (dit is een Sprite en de klasse Sprite breidt de klasse DisplayObjectContainer uit):

```
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);
```

De methode `getChildAt()` retourneert het onderliggende element dat zich op een opgegeven indexpositie in het weergaveoverzicht bevindt:

```
trace(container.getChildAt(0) is TextField); // true
```

U hebt ook toegang tot onderliggende objecten via de naam. Elk weergaveobject heeft een eigenschapnaam, en als u deze niet toewijst, wijst Flash Player of AIR een standaardwaarde toe, zoals `'instance1'`. De volgende code toont bijvoorbeeld hoe de methode `getChildByName()` moet worden gebruikt om toegang te verkrijgen tot een onderliggend weergaveobject met de naam `'banana loader'`:

```
trace(container.getChildByName("banana loader") is Loader); // true
```

Het gebruik van de methode `getChildByName()` resulteert mogelijk in langzamere prestaties dan wanneer u de methode `getChildAt()` zou gebruiken.

Omdat een weergaveobjectcontainer andere weergaveobjectcontainers als onderliggende objecten in zijn weergaveoverzicht kan bevatten, kunt u het volledige weergaveoverzicht van de toepassing als een boomstructuur doorlopen. In het voorgaande codefragment bevat het object `pict` één geladen onderliggend weergaveobject (de bitmap), zodra de laadbewerking van het Loader-object `pict` is voltooid. Als u toegang tot dit weergaveobject bitmap wilt, schrijft u `pict.getChildAt(0)`. U kunt ook `container.getChildAt(0).getChildAt(0)` schrijven, omdat `container.getChildAt(0) == pict`.

De volgende functie biedt de gewenste uitvoer `trace()` van het weergaveoverzicht van een weergaveobjectcontainer:

```
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
Als u Flex gebruikt, moet u weten dat Flex veel componenten weergaveobjectklassen definieert en dat deze klasse de toegangsmethoden tot het weergaveoverzicht van de klasse DisplayObjectContainer overschrijft. De klasse Container van het pakket mx.core overschrijft bijvoorbeeld de methode `addChild()` en andere methoden van de klasse DisplayObjectContainer (die de klasse Container uitbreidt). In het geval van de methode `addChild()` wordt de methode op een dergelijke manier door de klasse overschreven, dat u niet alle typen weergaveobjecten aan een instantie Container in Flex kunt toevoegen. De overschreven methode vereist in dit geval dat het onderliggende object dat u wilt toevoegen, een objecttype mx.core.UIComponent is.

De eigenschappen van Stage instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Stage overschrijft de meeste eigenschappen en methoden van de klasse DisplayObject. Als u een van deze overschreven eigenschappen of methoden aanroept, genereert Flash Player of AIR een uitzondering. Het object Stage heeft bijvoorbeeld niet de eigenschappen `x` of `y`, omdat zijn positie vast is als hoofdcontainer van de toepassing. De eigenschappen `x` en `y` verwijzen naar de positie van een weergaveobject ten opzichte van zijn container en omdat Stage zich niet in een andere weergaveobjectcontainer bevindt, zijn deze eigenschappen niet van toepassing.

Opmerking: *Sommige eigenschappen en methoden van de klasse Stage zijn alleen beschikbaar voor weergaveobjecten die zich in dezelfde beveiligingssandbox als het eerst geladen SWF-bestand. Zie "werkgebied, beveiliging" op pagina 1131 voor meer informatie.*

De framesnelheid voor het afspelen beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschap `framerate` van de klasse Stage wordt gebruikt om de framesnelheid voor alle SWF-bestanden die in de toepassing worden geladen in te stellen. Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Werkgebiedschaling beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer het formaat verandert van het gedeelte van het scherm dat Flash Player of AIR voorstelt, wordt de inhoud van het werkgebied automatisch door de runtime aangepast om dit te compenseren. De eigenschap `scaleMode` van de klasse Stage bepaalt hoe de inhoud van het werkgebied wordt aangepast. Deze eigenschap kan op vier verschillende waarden worden ingesteld, die als constanten zijn gedefinieerd in de klasse flash.display.StageScaleMode:

- `StageScaleMode.EXACT_FIT` schaalt het SWF-bestand om de nieuwe afmetingen van het werkgebied te vullen. Hierbij wordt geen rekening gehouden met de oorspronkelijke hoogte-breedteverhouding van de inhoud. Het is mogelijk dat de schaalfactoren voor de breedte en hoogte niet gelijk zijn, zodat de inhoud in elkaar geperst of juist uitgerekt wordt weergegeven wanneer de hoogte-breedteverhouding van het werkgebied wordt gewijzigd.
- `StageScaleMode.SHOW_ALL` schaalt het SWF-bestand zodanig dat dit volledig in de nieuwe afmetingen van het werkgebied past. Hierbij wordt de hoogte-breedteverhouding van de inhoud niet gewijzigd. In deze schaalmodus wordt alle inhoud weergegeven, maar kunnen er 'letterbox'-randen ontstaan, net als de zwarte balken die u ziet als u een breedbeeldfilm kijkt op een standaard-tv.

- `StageScaleMode.NO_BORDER` schaalt het SWF-bestand zodanig dat dit de nieuwe afmetingen van het werkgebied volledig vult. Hierbij wordt de hoogte-breedteverhouding van de inhoud niet gewijzigd. Bij deze schaalmodus wordt het volledige weergavegebied van het werkgebied benut, maar kan er inhoud worden afgesneden.
- `StageScaleMode.NO_SCALE` — het SWF-bestand wordt niet geschaald. Als de nieuwe afmetingen van het werkgebied kleiner zijn, wordt er inhoud afgesneden. Als de nieuwe afmetingen groter zijn, is de toegevoegde ruimte leeg.

  Alleen in de schaalmodus `StageScaleMode.NO_SCALE` kunnen de eigenschappen stage`Width` en stage`Height` van de klasse Stage worden gebruikt om de pixelafmetingen van de nieuwe werkgebiedgrootte te bepalen. (In de andere schaalmodi weerspiegelen de eigenschappen `stageWidth` en `stageHeight` altijd de oorspronkelijke breedte en hoogte van het SWF-bestand.) Wanneer `scaleMode` is ingesteld op `StageScaleMode.NO_SCALE` en de grootte van het SWF-bestand wordt aangepast, wordt bovendien de gebeurtenis `resize` van de klasse Stage verzonden, waardoor u dienovereenkomstig aanpassingen kunt maken.

  Omdat `scaleMode` is ingesteld op `StageScaleMode.NO_SCALE`, hebt u meer controle over hoe de scherminhoud aan de nieuwe grootte van het scherm wordt aangepast. In een SWF-bestand met een video en een bedieningsbalk, wilt u bijvoorbeeld dat de bedieningsbalk zijn grootte behoudt wanneer de grootte van het werkgebied wordt aangepast. Ook wilt u alleen de grootte van het videovenster wijzigen om de formaatwijziging van het werkgebied te compenseren. Dit wordt geïllustreerd in het volgende voorbeeld:

```
// 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;
}
```

De modus instellen voor werkgebiedschaling voor AIR-vensters

De werkgebiedeigenschap `scaleMode` bepaalt hoe onderliggende weergaveobjecten door het werkgebied worden geschaald en bijgesneden als het formaat van een venster wordt gewijzigd. In AIR mag alleen de modus `noScale` worden gebruikt. In deze modus wordt het werkgebied niet geschaald. In plaats hiervan verandert het formaat van het werkgebied direct mee met de begrenzingen van het venster. Objecten kunnen worden bijgesneden als het formaat van het venster kleiner wordt gemaakt.

De modi voor werkgebiedschaling zijn bedoeld voor omgevingen zoals webbrowsers, waarbij u het formaat of de hoogte-breedteverhouding van het werkgebied niet altijd zelf in de hand hebt. Met deze modi kunt u de minst slechte compromis kiezen wanneer het werkgebied niet overeenkomt met het ideale formaat of de ideale hoogte-breedteverhouding van uw toepassing. In AIR hebt u altijd de controle over het werkgebied, zodat u meestal betere resultaten krijgt als u de lay-out van de inhoud wijzigt of de venstergrootte aanpast in plaats van het werkgebied te schalen.

Voor de browser en het eerste AIR-venster worden gegevens over de relatie tussen het vensterformaat en de oorspronkelijke schaalfactor opgehaald uit het geladen SWF-bestand. Als u echter een NativeWindow-object maakt, kiest AIR een willekeurige relatie tussen het vensterformaat en de schaalfactor 72:1. Als uw venster een formaat heeft van 72 x 72 pixels en er wordt rechthoek van 10 x 10 aan het venster toegevoegd, wordt het venster dus getekend met het correcte formaat van 10 x 10 pixels. Heeft uw venster echter een formaat van 144 x 144 pixels, dan wordt een rechthoek van 10 x 10 pixels geschaald naar 20 x 20 pixels. Als voor de werkgebiedeigenschap `scaleMode` toch een andere waarde dan `noScale` wilt gebruiken, kunt u dit compenseren door de schaalfactor van willekeurige weergaveobjecten in het venster in te stellen op de verhouding van 72 pixels ten opzichte van de huidige breedte en hoogte van het werkgebied. In het volgende codevoorbeeld wordt de vereiste schaalfactor berekend voor een weergaveobject met de naam `client`:

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

Opmerking: *Bij Flex- en HTML-vensters wordt de werkgebiedeigenschap `scaleMode` automatisch ingesteld op `noScale`. Het wijzigen van de eigenschap `scaleMode` heeft bij dit soort vensters als gevolg dat de automatische lay-outmechanismen worden verstoord.*

Werken met de modus Volledig scherm

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de modus Volledig scherm kunt u instellen dat het werkgebied van een film het gehele scherm van de gebruiker vult, zonder randen of menu's. De eigenschap `displayState` van de klasse Stage wordt gebruikt om de modus Volledig scherm voor een SWF-bestand in en uit te schakelen. De eigenschap `displayState` kan op een van de waarden worden ingesteld, die zijn gedefinieerd door de constanten in de klasse flash.display.StageDisplayState. Als u de modus Volledig scherm wilt inschakelen, stelt u de eigenschap `displayState` in op `StageDisplayState.FULL_SCREEN`:

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Stel de eigenschap `displayState` in op `StageDisplayState.FULL_SCREEN_INTERACTIVE` om de interactieve schermvullende modus (nieuw in Flash Player 11.3) in te schakelen:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

In Flash Player kan de modus Volledig scherm alleen via ActionScript worden geïnitieerd als reactie op een muisklik (ook een rechtermuisklik) of een toetsdruk. In het geval van AIR-inhoud die wordt uitgevoerd in de sandbox met toepassingsbeveiliging hoeft de modus Volledig scherm niet te worden geactiveerd om te reageren op een beweging van de gebruiker.

Als u de modus Volledig scherm wilt uitschakelen, stelt u de eigenschap `displayState` in op `StageDisplayState.NORMAL`.

```
stage.displayState = StageDisplayState.NORMAL;
```

Een gebruiker kan de modus Volledig scherm verlaten door de focus naar een ander venster te verplaatsen of door een van de verschillende toetscombinaties te gebruiken: de Esc-toets (alle platforms), Ctrl-W (Windows), Cmd-W (Mac) of Alt-F4 (Windows).

De modus Volledig scherm inschakelen in Flash Player

Als u de modus Volledig scherm wilt inschakelen voor een SWF-bestand dat is ingesloten in een HTML-pagina, moet de HTML-code waarmee Flash Player moet worden ingesloten een tag `param` en een kenmerk `embed` met de naam `allowFullScreen` en de waarde `true` bevatten, zoals hier getoond:

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

In het Flash-ontwerpgereedschap selecteert u eerst Bestand -> Publicatie-instellingen en vervolgens de sjabloon Alleen Flash - volledig scherm toestaan op het tabblad HTML van het dialoogvenster Publicatie-instellingen.

In Flex zorgt u dat de HTML-sjabloon is voorzien van `<object>`- en `<embed>`-tags die volledig scherm ondersteunen.

Als u JavaScript in een webpagina gebruikt om tags voor SWF-insluiting te genereren, moet u het JavaScript wijzigen om de tag `allowFullScreen` en het kenmerk toe te voegen. Als uw HTML-pagina bijvoorbeeld de functie `AC_FL_RunContent()` gebruikt, (die wordt gebruikt in HTML-pagina's die worden gegenereerd door Flash Professional en Flash Builder), moet u als volgt de parameter `allowFullScreen` toevoegen aan die functieaanroep:

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

Dit geldt niet voor SWF-bestanden die met de zelfstandige Flash Player worden uitgevoerd.

Opmerking: *Als u de venstermodus (wmode in HTML) instelt op Dekkend, zonder venster (opaque) of Transparant, zonder venster (transparent), blijft het schermvullende venster altijd dekkend.*

Er gelden bovendien enkele beveiligingsbeperkingen voor het gebruik van de modus Volledig scherm met Flash Player in een browser. Deze beperkingen worden beschreven in "Beveiliging" op pagina 1106.

De interactieve schermvullende modus inschakelen in Flash Player 11.3 en later

Flash Player 11.3 en hoger biedt ondersteuning voor de interactieve schermvullende modus waarin alle toetsenbordtoetsen volledig worden ondersteund (met uitzondering van **Esc** waarmee deze modus wordt afgesloten). De interactieve schermvullende modus is handig tijdens het spelen van games, bijvoorbeeld om te kunnen chatten in een multiplayergame of voor de WASD-toetsenbordbesturingselementen in een FPS-game (First Person Shooter-game).

Als u de interactieve schermvullende modus wilt inschakelen voor een SWF-bestand dat is ingesloten in een HTML-pagina, moet de HTML-code voor het insluiten van Flash Player een tag `param` en een kenmerk `embed` met de naam `allowFullScreenInteractive` bevatten, plus de waarde `true`, zoals hier getoond:

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

In het Flash-ontwerpgereedschap selecteert u eerst Bestand -> Publicatie-instellingen en vervolgens de sjabloon Alleen Flash - volledig scherm toestaan op het tabblad HTML van het dialoogvenster Publicatie-instellingen.

Zorg dat de HTML-sjablonen in Flash Builder en Flex de tags`<object>` en `<embed>` bevatten ter ondersteuning van de interactieve schermvullende modus.

Als u JavaScript in een webpagina gebruikt om tags voor SWF-insluiting te genereren, moet u het JavaScript wijzigen om de tag en het kenmerk `allowFullScreenInteractive` toe te voegen. Als uw HTML-pagina bijvoorbeeld de functie `AC_FL_RunContent()` gebruikt (die wordt gebruikt in HTML-pagina's die worden gegenereerd door Flash Professional en Flash Builder), moet u als volgt de parameter `allowFullScreenInteractive` toevoegen aan die functieaanroep:

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

Dit geldt niet voor SWF-bestanden die met de zelfstandige Flash Player worden uitgevoerd.

Grootte van het werkgebied in het volledig scherm en schalen

De eigenschappen `Stage.fullScreenHeight` en `Stage.fullScreenWidth` retourneren de hoogte en breedte van de monitor die wordt gebruikt voor de modus Volledig scherm als deze direct wordt ingeschakeld. Deze waarden zijn mogelijk onjuist als de gebruiker de mogelijkheid heeft de browser van de ene naar de andere monitor te verplaatsen tussen het ophalen van de waarde en het overschakelen naar schermvullende weergave. Als u deze waarden ophaalt in dezelfde gebeurtenishandler als waarin u de eigenschap `Stage.displayState` instelt op `StageDisplayState.FULL_SCREEN`, zijn de waarden correct. Als een gebruiker meerdere schermen heeft, zal de inhoud van het SWF-bestand één scherm vullen. Flash Player en AIR gebruiken metrische informatie om te bepalen welke monitor het grootste gedeelte van het SWF-bestand bevat en gebruiken die monitor voor de modus Volledig scherm. De eigenschappen fullScreenHeight en fullScreenWidth geven alleen de afmetingen aan van de monitor die wordt gebruikt voor de modus Volledig scherm. Zie `Stage.fullScreenHeight` en `Stage.fullScreenWidth` in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Het gedrag van werkgebiedschaling in de modus Volledig scherm is hetzelfde als in de normale modus; het schalen wordt beheerd door de eigenschap `scaleMode` van de klasse Stage. Als de eigenschap `scaleMode` is ingesteld op `StageScaleMode.NO_SCALE`, weerspiegelen de eigenschappen `stageWidth` en `stageHeight` van het werkgebied de grootte van het schermgebied dat wordt gebruikt door het SWF-bestand (in dit geval het gehele scherm). Als het SWF-bestand wordt weergegeven in de browser, bepaalt de bijbehorende HTML-parameter de instelling.

U kunt de gebeurtenis `fullScreen` van de klasse Stage gebruiken om te detecteren en reageren wanneer de modus Volledig scherm is in- of uitgeschakeld. U wilt bijvoorbeeld items aan het scherm toevoegen of verwijderen, of de grootte van items wijzigen wanneer u de modus Volledig scherm in- of uitschakelt, zoals in dit voorbeeld:

```
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);
```

Zoals getoond door deze code, is het object Event voor de gebeurtenis `fullScreen` een instantie van de klasse flash.events.FullScreenEvent, die een eigenschap `fullScreen` bevat die aangeeft of de modus Volledig scherm is ingeschakeld (`true`) of niet (`false`).

Toetsenbordondersteuning in de modus Volledig scherm

Wanneer Flash Player wordt uitgevoerd in een browser, worden alle toetsenbordgerelateerde ActionScript-codes, zoals toetsenbordgebeurtenissen en tekstinvoer in instanties TextField uitgeschakeld in de modus Volledig scherm. De uitzonderingen (de toetsen die zijn ingeschakeld) zijn:

- Geselecteerde niet-afdrukbare toetsen, met name de pijltjestoetsen, spatiebalk en de tabtoets
- Sneltoetsen waarmee de modus Volledig scherm kan worden beëindigd: Esc (Windows en Mac), Ctrl-W (Windows), Command-W (Mac) en Alt-F4

Deze beperkingen gelden niet voor SWF-inhoud die wordt uitgevoerd in de zelfstandige versie van Flash Player of in AIR. AIR ondersteunt een interactieve schermvullende modus waarbij invoer met het toetsenbord is toegestaan.

Muisondersteuning in de modus Volledig scherm

Standaard functioneren muisgebeurtenissen in de modus Volledig scherm op dezelfde manier als in andere modi waarbij het scherm niet volledig wordt weergegeven. In de modus Volledig scherm, kunt u de eigenschap `Stage.mouseLock` echter ook zo instellen dat de muisvergrendeling optioneel kan worden ingeschakeld. Bij de muisvergrendeling wordt de cursor uitgeschakeld en kan de muis zonder begrenzingen worden verplaatst.

Opmerking: *U kunt de muisvergrendeling alleen inschakelen bij bureaubladtoepassingen in de modus Volledig scherm. Als u deze functie wilt instellen bij toepassingen die niet in de modus Volledig scherm zijn, of bij toepassingen op mobiele apparaten, treedt er een uitzonderingsfout op.*

Muisvergrendeling wordt automatisch uitgeschakeld en de muiscursor wordt weer zichtbaar als:

- De gebruiker de modus Volledig scherm afsluit door op Escape (alle platformen), Control-W (Windows), Command-W (Mac) of Alt-F4 (Windows) te drukken.
- Het toepassingsvenster verliest de focus.
- De instellingeninterface is zichtbaar, inclusief alle dialoogvensters voor privacy.
- Er wordt een native dialoogvenster weergegeven, bijvoorbeeld voor het uploaden van bestanden.

Gebeurtenissen die gekoppeld zijn aan muisbewegingen, zoals de gebeurtenis `mouseMove`, gebruiken de klasse MouseEvent om het gebeurtenisobject te representeren. Wanneer de muisvergrendeling is uitgeschakeld, gebruikt u de eigenschappen `MouseEvent.localX` en `MouseEvent.localY` om de locatie van de muis te bepalen. Wanneer de muisvergrendeling is ingeschakeld, gebruikt u de eigenschappen `MouseEvent.movementX` en `MouseEvent.movementY` om de locatie van de muis vast te stellen. De eigenschappen `movementX` en `movementY` bevatten gegevens over de positieveranderingen van de muis sinds de laatste gebeurtenis, in plaats van de absolute coördinaten van de muislocatie.

Hardwarematige schaling in de modus Volledig scherm

Met de eigenschap `fullScreenSourceRect` van de klasse Stage kunt u instellen dat Flash Player of AIR een bepaald gebied van het werkgebied schalen naar de modus Volledig scherm. Indien beschikbaar gebruiken Flash Player en AIR hardwarematige schaling, waarbij de videokaart op de computer van de gebruiker wordt gebruikt en inhoud doorgaans sneller wordt weergegeven dan bij softwarematige schaling.

U kunt gebruikmaken van hardwarematige schaling door het gehele werkgebied of een deel van het werkgebied in te stellen op de modus Volledig scherm. Met de volgende ActionScript 3.0-code wordt het gehele werkgebied ingesteld op de modus Volledig scherm:

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Wanneer deze eigenschap is ingesteld op een geldige rechthoek en de eigenschap `displayState` is ingesteld op de modus Volledig scherm, schaalt Flash Player of AIR het opgegeven gebied. De werkelijke werkgebiedgrootte in pixels wijzigt niet binnen ActionScript. Flash Player en AIR hanteren een minimumgrens voor de grootte van de rechthoek om het standaardbericht weer te geven waarin wordt aangegeven dat de gebruiker op Esc moet drukken om de modus Volledig scherm af te sluiten. Deze grens ligt meestal rond 260 x 30 pixels maar kan variëren, afhankelijk van het platform en de Flash Player-versie.

De eigenschap `fullScreenSourceRect` kan alleen worden ingesteld wanneer Flash Player of AIR niet in de modus Volledig scherm wordt uitgevoerd. Wanneer u deze eigenschap correct wilt gebruiken, moet u deze eerst instellen en vervolgens de eigenschap `displayState` op de modus Volledig scherm instellen.

U kunt schalen inschakelen door de eigenschap `fullScreenSourceRect` in te stellen op een rechthoekig object.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

U kunt schalen uitschakelen door de eigenschap `fullScreenSourceRect` in te stellen op `null`.

```
stage.fullScreenSourceRect = null;
```

Als u alle hardwareversnellingsfuncties van Flash Player wilt benutten, schakelt u deze in via het dialoogvenster met instellingen voor Flash Player. U kunt het dialoogvenster laden door met de rechtermuisknop (Windows) of terwijl u Control ingedrukt houdt (Mac) te klikken in Flash Player-inhoud in uw browser. Selecteer het tabblad Display, dit is het eerste tabblad, en klik op het selectievakje Hardwareversnelling.

Venstermodi Direct en GPU-compositing

Flash Player 10 introduceert twee venstermodi, Direct en GPU-compositing, die u kunt inschakelen via de publicatie-instellingen in het Flash-ontwerpgereedschap. Deze modi worden niet ondersteund in AIR. Als u deze modi wilt gebruiken, moet u de hardwareversnelling inschakelen voor Flash Player.

In de modus Direct wordt het snelste, meest directe pad gebruikt om grafische gegevens met push-technologie te leveren aan het scherm, wat gunstig is voor het afspelen van video.

GPU-compositing gebruikt de grafische verwerkingseenheid op de videokaart om de compositing te versnellen. Videocompositing is het samenvoegen van meerdere grafische lagen om één videobeeld te creëren. Als compositing wordt versneld met de GPU kunnen hierdoor de prestaties van YUV-conversie, kleurcorrectie, roteren of schalen, en overvloeien worden verbeterd. YUV-conversie is de kleurconversie van samengestelde analoge signalen die voor transmissie worden gebruikt, naar het kleurmodel RGB (rood, groen, blauw) dat voor het opnemen en afspelen van videobeelden wordt gebruikt. Door het gebruik van GPU om compositing te versnellen, is er minder geheugen nodig en hoeft de CPU minder berekeningen uit te voeren. Daarnaast wordt video in standaarddefinitie vloeiender afgespeeld.

Wees voorzichtig met het implementeren van deze venstermodi. Voor GPU-compositing zijn mogelijk veel geheugen- en CPU-resources vereist. Als sommige bewerkingen (zoals overvloeimodi, filteren, knippen en maskeren) niet kunnen worden uitgevoerd in de GPU, worden ze uitgevoerd door de software. Adobe raadt u aan niet meer dan één SWF-bestand per HTML-pagina uit te voeren wanneer u deze modi gebruikt, en deze modi niet in te schakelen voor banners. De functie Film testen van Flash maakt geen gebruik van hardwareversnelling maar u kunt hardwareversnelling gebruiken via de optie Voorvertoning publiceren.

Het instellen van een hogere framesnelheid in het SWF-bestand dan 60, de maximale snelheid voor schermvernieuwing, is zinloos. Door een framesnelheid tussen 50 en 55 in te stellen, worden verloren frames toegestaan die om diverse redenen soms optreden.

Voor de modus Direct is Microsoft® DirectX 9 vereist met 128 MB VRAM in Windows en OpenGL voor Apple Macintosh, Mac OS X v10.2 of hoger. Voor GPU-compositing zijn Microsoft DirectX 9 en ondersteuning voor Pixel Shader 2.0 vereist met 128 MB VRAM in Windows. Op Mac OS X en in Linux zijn voor GPU-compositing OpenGL 1.5 en diverse OpenGL-uitbreidingen vereist (framebufferobject, multitexture, shaderobjecten, arceringstaal, fragmentarcering).

U kunt de versnellingsmodi `direct` en `gpu` per SWF activeren in het dialoogvenster Publicatie-instellingen voor Flash. Ga hiervoor naar het menu Hardwareversnelling op het tabblad Flash. Als u Geen kiest, wordt de venstermodus `default`, `transparent` of `opaque` hersteld, zoals is vastgelegd in de instelling Venstermodus op het tabblad HTML.

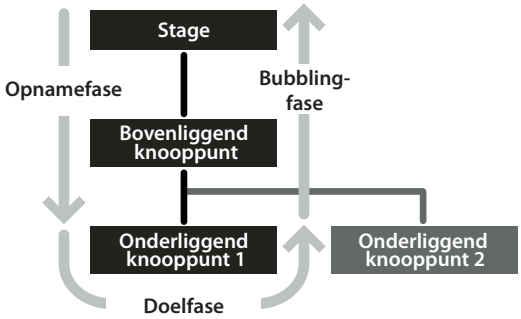
Gebeurtenissen voor weergaveobjecten afhandelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse DisplayObject overerft van de klasse EventDispatcher. Dit houdt in dat elk weergaveobject volledig kan deelnemen aan het gebeurtenismodel (dit wordt beschreven in "Gebeurtenissen afhandelen" op pagina 133). Elk weergaveobject kan zijn methode `addEventListener()` (die wordt overgeërfd van de klasse EventDispatcher) gebruiken om naar een bepaalde gebeurtenis te luisteren, maar alleen als het luisterende object deel is van de gebeurtenisstroom voor die gebeurtenis.

Wanneer Flash Player of AIR een object Event verzendt, maakt dit object Event een retourtocht van het werkgebied naar het weergaveobject waar de gebeurtenis heeft plaatsgevonden. Als een gebruiker bijvoorbeeld op een weergaveobject met de naam `child1` klikt, verzendt Flash Player een object Event van het werkgebied, via de hiërarchie van het weergaveoverzicht, naar beneden naar het weergaveobject `child1`.

De gebeurtenisstroom wordt in drie fasen verdeeld, zoals geïllustreerd in dit diagram:



Zie “Gebeurtenissen afhandelen” op pagina 133 voor meer informatie.

Wanneer u met weergaveobjectgebeurtenissen werkt, let dan op het effect dat de gebeurtenislisteners kunnen hebben op het feit of de weergaveobjecten automatisch uit het geheugen worden verwijderd (opschoning) wanneer ze uit het weergaveoverzicht worden verwijderd. Als een weergaveobject objecten bevat die als listeners zijn geabonneerd op deze gebeurtenissen, wordt dat weergaveobject niet uit het geheugen verwijderd als het uit het weergaveoverzicht wordt verwijderd. Het weergaveobject bevat namelijk nog steeds verwijzingen naar die listenerobjecten. Zie “Gebeurtenislisteners beheren” op pagina 147 voor meer informatie.

Een subklasse DisplayObject kiezen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat u kunt kiezen uit verschillende opties, is het bij het werken met weergaveobjecten heel belangrijk om na te gaan welk weergaveobject u voor een bepaald doel gaat gebruiken. De volgende richtlijnen vereenvoudigen de keuze. Dezelfde suggesties zijn van toepassing wanneer u een instantie van een klasse nodig hebt of als u een basisklasse kiest voor een klasse die u aan het maken bent:

- Als u geen object nodig hebt dat een container kan zijn voor overige weergaveobjecten (u hebt een object nodig dat moet dienen als zelfstandig schermelement), kiest u een van deze subklassen DisplayObject of InteractiveObject, afhankelijk van het gebruik:
  - Bitmap, voor het weergeven van een bitmapafbeelding.
  - TextField, voor het toevoegen van tekst.
  - Video, voor het weergeven van video.
  - Vorm voor een 'canvas' voor het tekenen van inhoud op het scherm. Als u bijvoorbeeld een instantie voor het tekenen van vormen op het scherm wilt maken en u wilt deze instantie niet gebruiken als container voor overige weergaveobjecten, dan kunt u de prestaties aanzienlijk verhogen door Shape te gebruiken in plaats van Sprite of MovieClip.
  - MorphShape, StaticText of SimpleButton voor items die met het Flash-programma voor het schrijven van programmacode zijn gemaakt. (U kunt met programmacode geen instanties van deze klassen maken, maar u kunt wel variabelen met deze gegevenstypen maken om naar items te verwijzen die zijn gemaakt met het Flash-programma voor het schrijven van programmacode.)
- Als u een variabele nodig hebt om naar het hoofdwerkgebied te verwijzen, gebruikt u de klasse Stage als het gegevenstype.

- Als u een container nodig hebt voor het laden van een extern SWF-bestand of afbeeldingsbestand, gebruikt u een instantie Loader. De geladen inhoud wordt aan het weergaveoverzicht toegevoegd als een onderliggend item van de instantie Loader. Het gegevenstype is afhankelijk van de aard van de geladen inhoud:
  - Een geladen afbeelding wordt een instantie Bitmap.
  - Een geladen SWF-bestand dat is geschreven in ActionScript 3.0 wordt een instantie Sprite of MovieClip (of een instantie van een subklasse van deze klassen; dit wordt bepaald door de maker van de inhoud).
  - Een geladen SWF-bestand dat is geschreven in ActionScript 1.0 of ActionScript 2.0 wordt een instantie AVM1Movie.
- Als het object functioneert als container voor overige weergaveobjecten (ongeacht of u met ActionScript op het weergaveobject tekent), kiest u een van de DisplayObjectContainer-subklassen:
  - Sprite, als het object alleen met ActionScript wordt gemaakt of als basisklasse voor een aangepast weergaveobject dat alleen met ActionScript wordt gemaakt en gemanipuleerd.
  - MovieClip, als u een variabele maakt die naar een filmclipsymbool moet verwijzen dat met het Flash-ontwerpgereedschap is gemaakt.
- Als u een klasse maakt die wordt gekoppeld aan een filmclipsymbool in de Flash-bibliotheek, kiest u een van deze DisplayObjectContainer-subklassen als de basisklasse van uw klasse:
  - MovieClip, als het gekoppelde filmclipsymbool op meer dan één frame inhoud heeft.
  - Sprite, als het gekoppelde filmclipsymbool alleen inhoud op het eerste frame heeft.

Weergaveobjecten manipuleren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Ongeacht het weergaveobject dat u kiest, bestaan er een aantal manipulaties die alle weergaveobjecten gemeen hebben, zoals elementen die op het scherm worden weergegeven. Deze kunnen bijvoorbeeld allemaal op het scherm worden gepositioneerd, naar voren of achter in de stapelvolgorde van weergaveobjecten worden verplaatst, worden geschaald en geroteerd, enzovoort. Omdat alle weergaveobjecten deze functies van de algemene basisklasse (DisplayObject) overerven, gedragen deze functies zich hetzelfde, ongeacht of u een instantie TextField, Video, Shape of overig weergaveobject manipuleert. In de volgende secties wordt dieper ingegaan op de verscheidene manipulaties van deze algemene weergaveobjecten.

De positie wijzigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De basismanipulatie van elk weergaveobject is de positionering ervan op het scherm. Als u de positie van een weergaveobject wilt instellen, wijzigt u de eigenschappen `x` en `y` van het object.

```
myShape.x = 17;
myShape.y = 212;
```

Het positioneringssysteem voor weergaveobjecten behandelt het werkgebied als een Cartesisch coördinatensysteem (het bekende rastersysteem met een horizontale x-as en een verticale y-as). De oorsprong van het coördinatensysteem (de 0,0-coördinaat waar de x en y samenkomen) bevindt zich in de linkerbovenhoek van het werkgebied. Vanaf dat punt zijn de x-waarden aan de rechterkant positief en aan de linkerkant negatief. In tegenstelling tot de typische grafieksystemen zijn de y-waarden naar beneden positief en naar boven negatief. De vorige coderegels verplaatsen het object `myShape` bijvoorbeeld naar x-coördinaat 17 (17 pixels naar rechts, vanaf de oorsprong) en y-coördinaat 212 (212 pixels onder de oorsprong).

Wanneer een weergaveobject wordt gemaakt met ActionScript, worden de eigenschappen `x` en `y` beide ingesteld op 0. Dit plaatst het object in de linkerbovenhoek van de bovenliggende inhoud.

De positie ten opzichte van het werkgebied wijzigen

Houd er rekening mee dat de eigenschappen `x` en `y` altijd naar de positie van het weergaveobject verwijzen ten opzichte van de 0,0-coördinaat van de bovenliggende assen van het weergaveobject. Voor een instantie Shape (zoals een cirkel) die zich in een instantie Sprite bevindt, betekent dit bijvoorbeeld dat als u de eigenschappen `x` en `y` van het object Shape op 0 instelt, de cirkel in de linkerbovenhoek van de Sprite wordt geplaatst (die mogelijk niet de linkerbovenhoek van het werkgebied is). Als u een object wilt positioneren ten opzichte van de algemene coördinaten van het werkgebied, kunt u de methode `globalToLocal()` van een weergaveobject als volgt gebruiken om algemene coördinaten (werkgebied) om te zetten in lokale coördinaten (weergaveobjectcontainer):

```
// 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;
```

U kunt nu op dezelfde manier de methode `localToGlobal()` van de klasse DisplayObject gebruiken om lokale coördinaten om te zetten in werkgebiedcoördinaten.

Weergaveobjecten met de muis verplaatsen

In ActionScript kunt u een gebruiker objecten laten verplaatsen aan de hand van twee verschillende technieken. In beide gevallen worden twee muisgebeurtenissen gebruikt. De ene gebeurtenis vindt plaats wanneer de muisknop wordt ingedrukt en het object de muiscursor volgt. De andere gebeurtenis vindt plaats wanneer de muisknop wordt losgelaten en de muiscursor niet langer door het object wordt gevolgd.

Opmerking: *Flash Player 11.3 en hoger; AIR 3.3 en hoger: u kunt de gebeurtenis MouseEvent.RELEASE_OUTSIDE ook toepassen voor het geval dat een gebruiker de muisknop loslaat buiten de begrenzingen van de omvattende Sprite.*

De eerste techniek, met gebruik van de methode `startDrag()`, is eenvoudiger maar heeft meer beperkingen. Wanneer de muisknop wordt ingedrukt, wordt de methode `startDrag()` van het te slepen weergaveobject aangeroepen. Wanneer de muisknop wordt losgelaten, wordt de methode `stopDrag()` aangeroepen. De klasse Sprite wordt door deze twee functies gedefinieerd, dus het verplaatste object moet deel uitmaken van de klasse Sprite of een subklasse hiervan.

```
// 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);
```

Deze methode heeft een belangrijke beperking: er kan slechts een item per keer worden gesleept met `startDrag()`. Als een weergaveobject wordt gesleept en de methode `startDrag()` voor een ander weergaveobject wordt aangeroepen, stopt het eerste weergaveobject direct met het volgen van de muis. Als de functie `startDragging()` bijvoorbeeld wordt gewijzigd, zoals hieronder aangegeven, wordt alleen het object `circle` gesleept, ondanks de methodeaanroep `square.startDrag()`:

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

Als gevolg van het feit dat er slechts een object per keer kan worden gesleept met `startDrag()`, kan de methode `stopDrag()` voor elk weergaveobject worden aangeroepen en stopt deze het slepen van het object dat op dat moment wordt gesleept.

Als u meer dan een weergaveobject moet slepen, of het risico op een conflict tussen verschillende objecten wilt vermijden die `startDrag()` gebruiken, is het raadzaam de muismethode te gebruiken om het sleepeffect te maken. Met deze techniek wordt, wanneer u de muisknop indrukt, een functie als listener geabonneerd op de gebeurtenis `mouseMove` van het werkgebied. Deze functie, die vervolgens elke keer dat de muis beweegt wordt aangeroepen, zorgt ervoor dat het te slepen object naar de x,y-coördinaten van de muis wordt verplaatst. Als de muisknop wordt losgelaten, wordt het abonnement van de functie als listener opgezegd. Dit houdt in dat deze niet langer wordt aangeroepen wanneer de muis wordt verplaatst en dat het object stopt met het volgen van de muiscursor. De volgende code demonstreert deze methode:

```
// 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);
```

Naast het feit dat de muiscursor wordt gevolgd door een weergaveobject, is het vaak gewenst dat het gesleepte object naar de voorgrond van het scherm wordt geplaatst zodat het lijkt of het boven de andere objecten drijft. Stel dat u beschikt over twee objecten, een cirkel en een vierkant, die allebei met de muis kunnen worden verplaatst. Als de cirkel zich in het weergaveoverzicht onder het vierkant bevindt en u op de cirkel klikt en deze sleept totdat de cursor zich boven het vierkant bevindt, wordt de cirkel achter het vierkant gesleept, waardoor de illusie van Slepen en neerzetten verdwijnt. In plaats hiervan kunt u ervoor zorgen dat wanneer op de cirkel wordt geklikt, de cirkel naar het hoogste niveau in het weergaveoverzicht wordt verplaatst en zodoende altijd boven de andere inhoud wordt weergegeven.

Met het volgende codevoorbeeld (een aanpassing van het vorige voorbeeld) kunnen twee weergaveobjecten, een cirkel en een vierkant, met de muis worden verplaatst. Wanneer boven een item op de muisknop wordt geklikt, wordt dat item naar het hoogste niveau van het weergaveoverzicht van het werkgebied verplaatst, zodat het gesleepte item altijd op de voorgrond wordt weergegeven. (Nieuwe of gewijzigde code ten opzichte van het vorige voorbeeld wordt vet weergegeven.)

```
// 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);
```

Als u dit effect wilt vergroten, zoals bijvoorbeeld voor een spel met stapels symbolen of kaarten, kunt u het object waarmee wordt gesleept toevoegen aan het weergaveoverzicht van het werkgebied wanneer deze wordt 'opgepakt' en het vervolgens aan een ander weergaveoverzicht toevoegen, zoals de 'stapel' waarop deze wordt losgelaten, op het moment dat de muisknop wordt losgelaten.

Ten slotte kunt u een filter Slagschaduw op het weergaveobject toepassen wanneer hierop wordt geklikt (wanneer u begint met slepen) en het filter opheffen wanneer het object wordt losgelaten. Zie “Weergaveobjecten filteren” op pagina 283 voor meer informatie over het gebruik van het filter Slagschaduw en overige filters voor weergaveobjecten in ActionScript.

Weergaveobjecten pannen en schuiven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een weergaveobject hebt dat te groot is voor het gebied waarin u het wilt weergeven, kunt u de eigenschap `scrollRect` gebruiken om het zichtbare gebied van het weergaveobject te definiëren. Als u de eigenschap `scrollRect` wijzigt als reactie op de gebruikersuitvoer, kunt u bovendien zorgen dat de inhoud naar links en rechts pant of naar boven en beneden schuift.

De eigenschap `scrollRect` is een instantie van de klasse Rectangle; dit is een klasse die de waarden combineert die nodig zijn om een rechthoekig gebied als een enkel object te definiëren. Als u het zichtbare gebied van het weergaveobject wilt definiëren, maakt u een nieuwe instantie Rectangle en wijst u deze toe aan de eigenschap `scrollRect` van het weergaveobject. Als u later wilt schuiven of pannen, leest u de eigenschap `scrollRect` in een aparte variabele Rectangle in, waarna u de gewenste eigenschap wijzigt (wijzig bijvoorbeeld de eigenschap `x` van de instantie Rectangle om te pannen of de eigenschap `y` om te schuiven). Vervolgens wijst u die instantie Rectangle toe aan de eigenschap `scrollRect` om de gewijzigde waarde aan het weergaveobject te melden.

Met de volgende code wordt bijvoorbeeld het zichtbare gebied gedefinieerd van een object TextField met de naam `bigText` dat te lang is om tussen de grenzen van het SWF-bestand te passen. Wanneer op de twee knoppen met de naam `up` en `down` wordt geklikt, roepen zij functies aan die ervoor zorgen dat de inhoud van het object TextField naar boven of naar beneden schuift door de eigenschap `y` van de Rectangle-instantie `scrollRect` te wijzigen.

```
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);
```

Dit voorbeeld demonstreert dat wanneer u met de eigenschap `scrollRect` van een weergaveobject werkt, het raadzaam is op te geven dat Flash Player of AIR de inhoud van het weergaveobject met behulp van de eigenschap `cacheAsBitmap` als bitmap in cache plaatst. Als u dit doet, hoeft Flash Player of AIR niet de gehele inhoud van het weergaveobject opnieuw te tekenen wanneer het object wordt verschoven en kan in plaats hiervan de bitmap in cache worden gebruikt om het benodigde gedeelte direct op het scherm te renderen. Zie "Weergaveobjecten in cache plaatsen" op pagina 193 voor meer informatie.

Grootte manipuleren en objecten schalen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de grootte van een weergaveobject op twee manieren meten en manipuleren, waarbij u de eigenschappen voor de grootte (`width` en `height`) of de eigenschappen voor het schalen (`scaleX` en `scaleY`) gebruikt.

Elk weergaveobject heeft een eigenschap `width` en `height`, die aanvankelijk zijn ingesteld op de grootte van het object in pixels. U kunt de waarden van die eigenschappen lezen om de grootte van het weergaveobject te meten. Ook kunt u als volgt nieuwe waarden opgeven om de grootte van het object te wijzigen:

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

Als u de eigenschappen `height` of `width` van een weergaveobject wijzigt, wordt het object hierdoor geschaald: de inhoud van het object wordt vergroot of verkleind om in het nieuwe gebied te passen. Als het weergaveobject vectorvormen bevat, worden deze vormen opnieuw getekend met de nieuwe schaal, zonder kwaliteitsverlies. Elk grafisch bitmapelement in het weergaveobject wordt geschaald en niet opnieuw getekend. Als de breedte en hoogte van bijvoorbeeld een digitale foto groter worden gemaakt dan de daadwerkelijke afmetingen van de pixelinformatie in de afbeelding, ziet de foto er korrelig en oneffen uit.

Wanneer u de eigenschap `width` of `height` van een weergaveobject wijzigt, werkt Flash Player of AIR de eigenschappen `scaleX` en `scaleY` van het object ook bij.

Opmerking: *TextField-objecten vormen een uitzondering op dit schalingsgedrag. Tekstvelden moeten hun eigen formaat veranderen ten behoeve van tekstomloop en tekengrootten; daarom worden hun scaleX- of scaleY-waarden weer op 1 ingesteld nadat het formaat is gewijzigd. Als u echter de scaleX- of scaleY-waarde van een TextField-object aanpast, veranderen de breedte- en hoogte-instellingen in overeenstemming met de schaalwaarden die u opgeeft.*

Deze eigenschappen vertegenwoordigen de relatieve grootte van het weergaveobject in vergelijking met de oorspronkelijke grootte. De eigenschappen `scaleX` en `scaleY` gebruiken breukwaarden (decimaal) om het percentage te vertegenwoordigen. Als de eigenschap `width` van een weergaveobject bijvoorbeeld zodanig is gewijzigd dat deze half zo groot is als het oorspronkelijke formaat, krijgt de eigenschap `scaleX` van het object de waarde `0,5` (50 procent). Als de hoogte is verdubbeld, krijgt de eigenschap `scaleY` de waarde `2` (200 procent).

```
// 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
```

Formaatwijzigingen zijn niet proportioneel. Met andere woorden, als u van een vierkant de `height` wijzigt, maar niet de `width`, zijn deze eigenschappen niet meer gelijk en is er sprake van een rechthoek in plaats van een vierkant. Als u de grootte van een weergaveobject proportioneel wilt wijzigen, kunt u de waarden van de eigenschappen `scaleX` en `scaleY` instellen, als alternatief voor de eigenschappen `width` of `height`. Deze code wijzigt bijvoorbeeld de `width` van het weergaveobject met de naam `square`. Vervolgens wordt de verticale schaal (`scaleY`) aangepast aan de horizontale schaal, waardoor de zijden van het vierkant gelijk blijven.

```
// 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;
```

De vervorming tijdens het schalen beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer een weergaveobject wordt geschaald (bijvoorbeeld een horizontale vergroting), wordt de resulterende vervorming in gelijke mate over het object verspreid, zodat ieder deel met dezelfde hoeveelheid wordt vergroot. Voor afbeeldingen en ontwerpelementen is dit de gewenste bewerking. Soms wilt u echter meer controle over welk gedeelte van het weergaveobject u wilt vergroten en welk gedeelte u niet wilt wijzigen. Een voorbeeld hiervan is een rechthoekige knop met afgeronde hoeken. Bij een normale schaling worden de hoeken van de knop vergroot, waardoor de hoekstraal wordt gewijzigd als het formaat van de knop wordt aangepast.

In dit geval is meer controle over het schalen dus gewenst. U kunt hiermee bepaalde gebieden aanduiden die moeten worden geschaald (de rechte zijden en het midden) en de gebieden die niet mogen worden geschaald (de hoeken), zodat het schalen wordt uitgevoerd zonder zichtbare vervorming.

U kunt een 9-delige schaling (schaal 9) gebruiken om weergaveobjecten te maken. Hierdoor kunt u beter aangeven hoe de objecten moeten worden geschaald. Met een 9-delige schaling wordt het weergaveobject in negen aparte rechthoeken verdeeld (een raster van 3 x 3, zoals het raster van een boter-kaas-en-eieren-bord). De rechthoeken hebben niet noodzakelijkerwijs hetzelfde formaat, u kunt aangeven waar de rasterlijnen moeten worden geplaatst. De inhoud die zich in de vier rechthoeken in de hoeken bevindt (zoals de afgeronde hoeken van een knop) worden niet vergroot of verkleind wanneer het weergaveobject wordt geschaald. De middelste rechthoeken aan de bovenkant en onderkant worden horizontaal (maar niet verticaal) geschaald, terwijl de middelste rechthoeken aan de linkerkant en rechterkant verticaal (maar niet horizontaal) worden geschaald. De rechthoek in het midden wordt zowel horizontaal als verticaal geschaald.

Houd dit in gedachten wanneer u een weergaveobject maakt en u wilt dat bepaalde inhoud niet wordt geschaald; u hoeft er alleen voor te zorgen dat de scheidende lijnen van het raster voor 9-delige schaling zo worden geplaatst dat de inhoud in een van de rechthoeken in de hoeken terechtkomt.

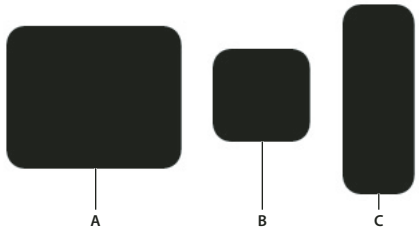
Als u in ActionScript een waarde instelt voor de eigenschap `scale9Grid` van een weergaveobject, wordt 9-delige schaling voor het object ingesteld en wordt de grootte van elk rechthoek in het 9-delige raster voor het object gedefinieerd. U kunt als volgt een instantie van de klasse Rectangle gebruiken als waarde voor de eigenschap `scale9Grid`:

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

De vier parameters van de constructor Rectangle zijn de x-coördinaat, de y-coördinaat, de breedte en de hoogte. In dit voorbeeld wordt de linkerbovenhoek van de rechthoek op punt x: 32, y: 27 geplaatst, op een weergaveobject met de naam `myButton`. De rechthoek is 71 pixels breed en 64 pixels hoog (de rechterzijde bevindt zich op x-coördinaat 103 van het weergaveobject en de onderste rand bevindt zich op y-coördinaat 92 van het weergaveobject).

Het daadwerkelijke gebied binnen het gebied dat wordt gedefinieerd door de instantie Rectangle, vertegenwoordigt de middelste rechthoek van een raster voor 9-delige schaling. De overige rechthoeken worden door Flash Player of AIR berekend door de zijden van de instantie Rectangle te vergroten, zoals hier getoond:

Wanneer de knop groter of kleiner wordt geschaald, worden de afgeronde hoeken in dit geval niet vergroot of verkleind, maar worden de overige gebieden wel aangepast.



A. *myButton.width = 131;myButton.height = 106;* ***B.*** *myButton.width = 73;myButton.height = 69;* ***C.*** *myButton.width = 54;myButton.height = 141;*

Weergaveobjecten in cache plaatsen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naarmate uw ontwerpen in Flash groter worden, moet u nadenken over prestaties en optimalisatie, of u nu een toepassing of complexe gescripte animaties maakt. Wanneer u inhoud hebt die statisch blijft (zoals een rechthoekige Shape-instantie), optimaliseren Flash Player en AIR de inhoud niet. Wanneer u de positie van de rechthoek wijzigt, wordt de gehele Shape-instantie opnieuw in Flash Player of AIR getekend.

U kunt opgegeven objecten in cache plaatsen om de prestaties van het SWF-bestand te verbeteren. Het weergaveobject is een *oppervlak*, in feite een bitmapversie van de vectorgegevens van de instantie. U zult deze gegevens tijdens het verloop van het SWF-bestand waarschijnlijk niet veel willen wijzigen. Instanties waarvoor plaatsing in cache is ingeschakeld, worden dus niet continu opnieuw getekend tijdens het afspelen van het SWF-bestand, waardoor het SWF-bestand sneller kan worden gerenderd.

Opmerking: *U kunt de vectorgegevens bijwerken, waarbij het oppervlak opnieuw wordt gemaakt. De vectorgegevens die in cache zijn geplaatst, hoeven hierdoor niet voor het hele SWF-bestand gelijk te blijven.*

Als u de eigenschap `cacheAsBitmap` van een weergaveobject op `true` instelt, zorgt u ervoor dat het weergaveobject een bitmaprepresentatie van zichzelf in cache plaatst. Flash Player of AIR maakt een oppervlakobject voor de instantie. Dit is een in cache geplaatste bitmap in plaats van vectorgegevens. Wanneer u de grenzen van het weergaveobject wijzigt, wordt het oppervlak opnieuw gemaakt (en niet alleen maar gewijzigd). Oppervlakken kunnen binnen andere oppervlakken worden genest. De bitmap van het onderliggende oppervlak wordt naar het bovenliggende oppervlak gekopieerd. Zie "Bitmap in cache plaatsen" op pagina 196 voor meer informatie.

De eigenschappen `opaqueBackground` en `scrollRect` van de klasse DisplayObject zijn gerelateerd aan de eigenschap `cacheAsBitmap`, waarbij een bitmap in cache wordt geplaatst. Hoewel deze drie eigenschappen onafhankelijk van elkaar zijn, werken de eigenschappen `opaqueBackground` en `scrollRect` het best wanneer een object als bitmap in cache wordt geplaatst. De prestaties van de eigenschappen `opaqueBackground` en `scrollRect` verbeteren alleen wanneer u `cacheAsBitmap` instelt op `true`. Zie "Weergaveobjecten pannen en schuiven" op pagina 189 voor meer informatie over het schuiven van inhoud van weergaveobjecten. Zie "Een dekkende achtergrondkleur instellen" op pagina 196 voor meer informatie over het instellen van een dekkende achtergrond.

Zie `Weergaveobjecten maskeren` voor meer informatie over alfkanaalmaskering, waarvoor u de eigenschap `cacheAsBitmap` op "Weergaveobjecten maskeren" op pagina 201 moet instellen.

Wanneer moet u plaatsing in cache inschakelen?

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer plaatsing in cache is ingeschakeld voor een weergaveobject, wordt een oppervlak gemaakt. Dit biedt een aantal voordelen: zo worden complexe vectoranimaties bijvoorbeeld snel gerenderd. Er zijn diverse scenario's denkbaar waarin het inschakelen van plaatsing in cache nuttig is. Misschien bent u geneigd plaatsing in cache altijd in te schakelen om de prestaties van SWF-bestanden te verbeteren. In sommige situaties worden de prestaties echter niet verbeterd bij plaatsing in cache, de prestaties kunnen zelfs verslechteren. In deze sectie worden scenario's beschreven waarbij plaatsing in cache moet worden gebruikt en er wordt beschreven wanneer u beter standaardweergaveobjecten kunt gebruiken.

De totale prestaties van gegevens die in cache zijn geplaatst, zijn afhankelijk van de complexiteit van de vectorgegevens van uw instanties, van de hoeveelheid gegevens die u wijzigt en van de instelling van de eigenschap `opaqueBackground`. Wanneer u kleine gebieden wijzigt, kan het verschil tussen het gebruik van een oppervlak en vectorgegevens verwaarloosbaar klein zijn. Het kan zinvol zijn beide scenario's voor uw werk uit te proberen voordat u de toepassing implementeert.

Wanneer moet u bitmaps in cache plaatsen?

Hier volgen typische scenario's waarin plaatsing van bitmap in cache aanzienlijke voordelen kan opleveren.

- Complexe achtergrondafbeelding: een toepassing die een gedetailleerde en complexe achtergrondafbeelding van vectorgegevens bevat (bijvoorbeeld een afbeelding waarin u de opdracht Bitmap overtrekken hebt gebruikt of illustraties die u hebt gemaakt in Adobe Illustrator®). U kunt tekens laten bewegen tegen een achtergrond, waardoor de animatie wordt vertraagd, omdat de achtergrond de vectorgegevens continu opnieuw moet genereren. Als u de prestaties wilt verbeteren, kunt u de eigenschap `opaqueBackground` van het weergaveobject op de achtergrond instellen op `true`. De achtergrond wordt gerenderd als een bitmap en kan snel opnieuw worden getekend, waardoor de animatie veel sneller wordt afgespeeld.

- Schuivend tekstveld: een toepassing waarin een grote hoeveelheid tekst in een schuivend tekstveld wordt weergegeven. U kunt het tekstveld in een weergaveobject plaatsen die u instelt als schuifbaar met verschuivende grenzen (de eigenschap `scrollRect`). Hiermee wordt snelle pixelverschuiving ingeschakeld voor de opgegeven instantie. Wanneer een gebruiker de weergaveobjectinstantie schuift, worden de verschoven pixels omhoog verplaatst en wordt het nieuwe weergegeven gebied gegenereerd zonder het hele tekstveld opnieuw te genereren.
- Venstersysteem: een toepassing met een complex systeem van overlappende vensters. Elk venster (zoals vensters van een webbrowser) kan zijn geopend of gesloten. Wanneer u elk venster als een oppervlak markeert (door de eigenschap `cacheAsBitmap` op `true` in te stellen), wordt elk venster van de rest gescheiden en in cache geplaatst. Gebruikers kunnen de vensters slepen, zodat deze elkaar overlappen. Op die manier hoeft de vectorinhoud niet voor elk venster opnieuw te worden gegenereerd.
- Alpha-kanaalmaskering: wanneer u met alpha-kanaalmaskering werkt, moet u de eigenschap `cacheAsBitmap` instellen op `true`. Zie "Weergaveobjecten maskeren" op pagina 201 voor meer informatie.

Als u in al deze scenario's de bitmap in cache plaatst, wordt de reactiesnelheid en de interactiviteit van de toepassing verbeterd dankzij optimalisatie van de vectorafbeeldingen.

Als u bovendien een filter toepast op een weergaveobject, wordt `cacheAsBitmap` automatisch op `true` ingesteld, zelfs als u deze eigenschap handmatig op `false` hebt ingesteld. Wanneer u alle filters voor een weergaveobject wist, wordt de eigenschap `cacheAsBitmap` gewijzigd in de laatst ingestelde waarde.

Wanneer moet u bitmapcaching vermijden?

Het gebruik van deze functie in de onjuiste situatie heeft ongewenste invloed op de prestaties van uw SWF-bestand. Gebruik de volgende richtlijnen wanneer u bitmaps in cache plaatst:

- Maak geen overmatig gebruik van oppervlakken (weergaveobjecten waarvoor caching is ingeschakeld). Elk oppervlak gebruikt meer geheugen dan een standaardweergaveobject. U moet oppervlakken dan ook alleen inschakelen wanneer u renderingprestaties moet verbeteren.

  Een in cache geplaatste bitmap kan aanzienlijk meer geheugen gebruiken dan een standaardweergaveobject. Wanneer een instantie Sprite in het werkgebied 250 x 250 pixels groot is, kan de instantie bij plaatsing in cache bijvoorbeeld 250 kB in beslag nemen. Bij een standaardinstantie Sprite (niet in cache geplaatst) is dit 1 kB.
- Vermijd in- en uitzoomen op oppervlakken die in cache zijn geplaatst. Wanneer u te veel gebruikmaakt van bitmapcaching, wordt zeer veel geheugen gebruikt (zie vorige punt), met name wanneer u inzoomt op de inhoud.
- Gebruik oppervlakken voor weergaveobjectinstanties die overwegend statisch (zonder animatie) zijn. U kunt de instantie slepen of verplaatsen, maar de inhoud van de instantie mag niet bewegen of veel veranderen. (Animatie of bewegende inhoud wordt veelal gebruikt met een instantie MovieClip die animatie bevat of een instantie Video.) Wanneer u bijvoorbeeld een instantie roteert of transformeert, verandert de instantie tussen het oppervlak en de vectorgegevens. De verwerking hiervan is moeilijk en heeft een negatief effect op het SWF-bestand.
- Wanneer u oppervlakken combineert met vectorgegevens, neemt de hoeveelheid verwerking voor Flash Player of AIR (en soms voor de computer) toe. Groepeer oppervlakken zoveel mogelijk, bijvoorbeeld wanneer u venstertoepassingen maakt.
- Plaats geen objecten waarvan de afbeeldingen regelmatig veranderen in de cache. Steeds als u het weergaveobject schaalt, schuintrekt of roteert, de alfa- of kleurtransformatie wijzigt, onderliggende weergaveobjecten verplaatst of tekent met gebruik van de graphics-eigenschap, wordt de cachebitmap opnieuw getekend. Als dit in elk frame gebeurt, moet de runtime het object in een bitmap tekenen en deze bitmap vervolgens kopiëren naar het werkgebied. Dat is meer werk dan alleen het tekenen van het object buiten de cache naar het werkgebied. De nadelige invloed op de prestaties van opnemen in cache in vergelijking met de bijwerkfrequentie is afhankelijk van de complexiteit en grootte van het weergaveobject en kan alleen worden bepaald door het testen van de specifieke inhoud.

Bitmap in cache plaatsen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een weergaveobject als bitmap in cache wilt plaatsen, stelt u de eigenschap `cacheAsBitmap` in op `true`:

```
mySprite.cacheAsBitmap = true;
```

Nadat u de eigenschap `cacheAsBitmap` hebt ingesteld op `true`, verspringt het weergaveobject automatisch naar hele pixelcoördinaten. Wanneer u het SWF-bestand test, worden de animaties die worden uitgevoerd op een complexe vectorafbeelding, merkbaar sneller gerenderd.

In de volgende situaties wordt een oppervlak (in cache geplaatste bitmap) niet gemaakt, zelfs niet wanneer `cacheAsBitmap` is ingesteld op `true`:

- De bitmap is meer dan 2880 pixels hoog of breed.
- De bitmap kan niet worden toegewezen (fout als gevolg van onvoldoende geheugen).

Transformatiematrices voor bitmaps in cache

Adobe AIR 2.0 en hoger (mobiel profiel)

In AIR-toepassingen voor mobiele apparatuur dient u de eigenschap `cacheAsBitmapMatrix` altijd in te stellen als u de eigenschap `cacheAsBitmap` instelt. Zo kunt u een groter scala aan transformaties toepassen op het weergaveobject zonder rendering te activeren.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

Wanneer u deze matrixeigenschap instelt, kunt u de volgende aanvullende transformaties toepassen op het weergaveobject zonder dat dit opnieuw in de cache wordt geplaatst:

- Verplaatsen of vertalen zonder pixels uit te lijnen
- Roteren
- Schalen
- Scheeftrekken
- Alfa wijzigen (tussen 0 en 100% transparantie)

Deze transformaties worden rechtstreeks toegepast op de cachebitmap.

Een dekkende achtergrondkleur instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een dekkende achtergrond instellen voor een weergaveobject. Wanneer uw SWF-bestand een achtergrond heeft die een complexe vectortekening bevat, kunt u de eigenschap `opaqueBackground` bijvoorbeeld instellen op een opgegeven kleur (doorgaans dezelfde kleur als het werkgebied). De kleur moet als een getal worden opgegeven (meestal een hexadecimale kleurwaarde). De achtergrond wordt dan behandeld als een bitmap, wat optimalisatie van de prestaties bevordert.

Wanneer u `cacheAsBitmap` instelt op `true` en de eigenschap `opaqueBackground` instelt op een opgegeven kleur, kan de interne bitmap via de eigenschap `opaqueBackground` dekkend worden gemaakt en sneller worden gerenderd. Wanneer u `cacheAsBitmap` niet instelt op `true`, wordt met de eigenschap `opaqueBackground` een dekkende vectorrechthoek toegevoegd aan de achtergrond van het weergaveobject. Er wordt niet automatisch een bitmap gemaakt.

In het volgende voorbeeld ziet u hoe u de achtergrond van een weergaveobject kunt instellen om prestaties te optimaliseren:

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

In dit geval wordt de achtergrondkleur van Shape met de naam `myShape` ingesteld op rood (`0xFF0000`). Als de instantie Shape bijvoorbeeld een tekening bevat van een groene driehoek in een werkgebied met een witte achtergrond, wordt een groene driehoek weergegeven met rood in de lege ruimte van het selectiekader van de instantie Shape (de rechthoek die het werkgebied omvat).

Deze code heeft vanzelfsprekend meer nut als het werkgebied een effen rode achtergrond heeft. Als de achtergrond een andere kleur heeft, wordt deze kleur opgegeven. Als het een SWF-bestand met een witte achtergrond betreft, wordt de eigenschap `opaqueBackground` hoogstwaarschijnlijk ingesteld op `0xFFFFFF` (puur wit).

Overvloeimodi toepassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij overvloeimodi worden de kleuren van een afbeelding (de basisafbeelding) gecombineerd met de kleuren van een andere afbeelding (de overvloeiafbeelding). De resulterende, derde afbeelding wordt op het scherm weergegeven. Elke pixelwaarde in de basisafbeelding wordt verwerkt met de bijbehorende pixelwaarde van de andere afbeelding om een pixelwaarde te produceren voor diezelfde positie in de resulterende afbeelding.

Elk weergaveobject bevat een eigenschap `blendMode` die op een van de volgende overvloeimodi kan worden ingesteld. Dit zijn constanten die in de klasse BlendMode zijn gedefinieerd. U kunt ook de tekenreekswaarden (tussen ronde haakjes) gebruiken. Dit zijn de huidige waarden van de constanten.

- `BlendMode.ADD` (`"add"`): veelal gebruikt om een bewegend oplichtend verspreidingseffect te maken tussen twee afbeeldingen.
- `BlendMode.ALPHA` (`"alpha"`): veelal gebruikt om de transparantie van de voorgrond toe te passen op de achtergrond. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.DARKEN` (`"darken"`): veelal gebruikt voor overlappende tekst. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.DIFFERENCE` (`"difference"`): veelal gebruikt om levendigere kleuren te maken.
- `BlendMode.ERASE` (`"erase"`): veelal gebruikt om een gedeelte van de achtergrond uit te knippen (te wissen) met gebruikmaking van de voorgrond-alpha. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.HARDLIGHT` (`"hardlight"`): veelal gebruikt om schaduweffecten te maken. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.INVERT` (`"invert"`): gebruikt om de achtergrond om te keren.
- `BlendMode.LAYER` (`"layer"`): gebruikt om het maken van een tijdelijke buffer af te dwingen voor het vooraf samenstellen van een bepaald weergaveobject. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.LIGHTEN` (`"lighten"`): veelal gebruikt voor overlappende tekst. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.MULTIPLY` (`"multiply"`): veelal gebruikt om schaduw- en diepte-effecten te maken.

- `BlendMode.NORMAL` (`"normal"`): gebruikt om aan te geven dat de pixelwaarden voor de overvloeiafbeelding de pixelwaarden van de basisafbeelding overschrijven.
- `BlendMode.OVERLAY` (`"overlay"`): veelal gebruikt om schaduweffecten te maken. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.SCREEN` (`"screen"`): veelal gebruikt om markeringen en zonlichteffecten te maken.
- `BlendMode.SHADER` (`"shader"`): gebruikt om aan te geven dat er een Pixel Bender-arcering wordt gebruikt om een aangepast overvloei-effect te realiseren. Zie "Werken met Pixel Bender-arceringen" op pagina 318 voor meer informatie over het werken met shaders. (Niet ondersteund bij GPU-rendering.)
- `BlendMode.SUBTRACT` (`"subtract"`): veelal gebruikt om een bewegend verdonkerend verspreidingseffect te maken tussen twee afbeeldingen.

Kleuren DisplayObject wijzigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de methoden van de klasse ColorTransform (flash.geom.ColorTransform) gebruiken om de kleur van een weergaveobject aan te passen. Elk weergaveobject bevat een eigenschap `transform`. Dit is een instantie van de klasse Transform die informatie bevat over de verschillende transformaties die op het weergaveobject worden toegepast (zoals roteren, schaalwijzigingen enzovoort). Naast de informatie over geometrische transformaties, bevat de klasse Transform ook de eigenschap `colorTransform`. Dit is een instantie van de klasse ColorTransform die ervoor zorgt dat u de kleur van het weergaveobject kunt wijzigen. Als u de gegevens over de kleurtransformatie van een weergaveobject wilt bekijken, kunt u bijvoorbeeld de volgende code gebruiken:

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

Wanneer u een instantie ColorTransform hebt gemaakt, kunt u zijn eigenschappen lezen om te zien welke kleurtransformaties al zijn toegepast, of u kunt deze waarden instellen om kleurwijzigingen op het weergaveobject toe te passen. Als u het weergaveobject wilt bijwerken nadat u wijzigingen hebt aangebracht, moet u de instantie ColorTransform opnieuw aan de eigenschap `transform.colorTransform` toewijzen.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

De nieuwe kleurwaarden via code instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschap `color` van de klasse ColorTransform kan worden gebruikt om een specifieke rode, groene of blauwe (RGB) kleurwaarde aan het weergaveobject toe te wijzen. Het volgende voorbeeld gebruikt de eigenschap `color` om de kleur van het weergaveobject met de naam `square` in blauw te wijzigen op het moment dat de gebruiker op een knop met de naam `blueBtn` klikt:

```
// 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);
```

Wanneer u de kleur van een weergaveobject wijzigt via de eigenschap `color`, wordt de kleur van het gehele object gewijzigd, ongeacht of het object hiervoor meerdere kleuren had. Als u bijvoorbeeld een weergaveobject gebruikt dat een groene cirkel met zwarte tekst bevat, wordt het gehele object, de gehele cirkel en alle tekst rood wanneer u de eigenschap `color` van de gekoppelde instantie ColorTransform van dat object op een rode tint instelt (tekst en object zijn niet langer te onderscheiden).

Kleur- en helderheideffecten met code wijzigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Stel dat u een weergaveobject met meerdere kleuren hebt (bijvoorbeeld een digitale foto) en dat u niet het gehele object een nieuwe kleur wilt geven. U wilt slechts de kleur van een weergaveobject op basis van de bestaande kleuren wijzigen. Voor dit scenario bevat de klasse ColorTransform verschillende vermenigvuldigings- en verschuivingseigenschappen die u kunt gebruiken om deze wijziging te bewerkstelligen. De vermenigvuldigingseigenschappen (`redMultiplier`, `greenMultiplier`, `blueMultiplier` en `alphaMultiplier`) werken op dezelfde manier als gekleurde fotografische filters (of een zonnebril met gekleurde glazen), ze verhogen of verlagen bepaalde kleuren van het weergaveobject. De verschuivingseigenschappen (`redOffset`, `greenOffset`, `blueOffset` en `alphaOffset`) kunnen worden gebruikt om extra hoeveelheden van een bepaalde kleur aan het object toe te voegen of om de minimale waarde van een bepaalde kleur op te geven.

Deze vermenigvuldigings- en verschuivingseigenschappen en de geavanceerde kleurinstellingen die beschikbaar zijn voor filmclipsymbolen in het Flash-ontwerpgereedschap zijn identiek wanneer u Geavanceerd in het pop-upmenu Kleur in Eigenschapcontrole selecteert.

De volgende code laadt een JPEG-afbeelding en past een kleurtransformatie toe waarmee de rood- en groenkanalen worden aangepast terwijl de muisaanwijzer beweegt langs de x- en y-as. Omdat er geen verschuivingswaarden zijn opgegeven, is de kleurwaarde van elk kleurkanaal dat op het scherm wordt weergegeven een percentage van de oorspronkelijke kleurwaarde in de afbeelding. Dit houdt in dat de meeste hoeveelheid rood of groen die in een pixel wordt weergegeven overeenkomt met de oorspronkelijke hoeveelheid rood of groen in die 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);
```

Objecten roteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Weergaveobjecten kunnen worden geroteerd met de eigenschap `rotation`. Deze waarde geeft aan of een object al dan niet is geroteerd. Als u het object wilt roteren, kunt u deze eigenschap instellen op een getal (in graden) waarmee u aangeeft in welke mate het object moet worden geroteerd. Met de volgende coderegel wordt het object met de naam `square` bijvoorbeeld geroteerd met 45 graden (1/8 van een volledige draai):

```
square.rotation = 45;
```

Zie “Werken met geometrie” op pagina 223 om een weergaveobject te roteren met gebruik van een transformatiematrix.

Objecten faden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de transparantie van een weergaveobject beheren en het object gedeeltelijk of volledig transparant maken. U kunt ook de transparantie wijzigen om het object in of uit te laten faden. De eigenschap `alpha` van de klasse DisplayObject definieert de transparantie (of meer bepaald, de dekking) van een weergaveobject. De eigenschap `alpha` kan op een waarde tussen 0 en 1 worden ingesteld, waar 0 gelijk staat aan volledige transparantie en 1 aan volledige dekking. Deze coderegels maken het object met de naam `myBall` bijvoorbeeld gedeeltelijk (50 procent) transparant op het moment dat de gebruiker met de muis op het object klikt:

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

U kunt de transparantie van een weergaveobject ook wijzigen met gebruik van de kleuraanpassingen die beschikbaar zijn via de klasse ColorTransform. Zie “Kleuren DisplayObject wijzigen” op pagina 198 voor meer informatie.

Weergaveobjecten maskeren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een weergaveobject ook als een masker gebruiken om een gat te maken waardoor de inhoud van een ander weergaveobject zichtbaar is.

Een masker definiëren

Als u wilt aangeven dat een weergaveobject functioneert als masker voor een ander weergaveobject, stelt u het maskeerobject in als de eigenschap `mask` van het weergaveobject dat moet worden gemaskeerd:

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

Het gemaskeerde weergaveobject wordt onthuld onder alle dekkende (niet-transparante) gebieden van het weergaveobject dat als masker fungeert. De volgende code maakt bijvoorbeeld een instantie Shape met een vierkant van 100 x 100 pixels en een instantie Sprite met een blauwe cirkel met een straal van 25 pixels. Wanneer op de cirkel wordt geklikt, wordt de cirkel ingesteld als masker voor het vierkant, zodat het enige zichtbare gedeelte van het vierkant het gedeelte is dat wordt bedekt door het effen gedeelte van de cirkel. Uiteindelijk is er dus alleen een rode cirkel zichtbaar.

```
// 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);
```

Het weergaveobject dat als een masker fungeert, kan sleepbaar zijn en bewegen. Hiernaast kan de grootte van het object dynamisch worden aangepast en kan het object afzonderlijke vormen binnen een enkel masker gebruiken. Het maskerende weergaveobject hoeft niet noodzakelijkerwijs aan het weergaveoverzicht te worden toegevoegd. Als u echter wilt dat het maskerende object schaalt wanneer het werkgebied wordt geschaald of als u gebruikersinteractie met het masker wilt inschakelen (zoals slepen en formaatwijzigingen door de gebruiker), moet het maskerende object aan het weergaveoverzicht worden toegevoegd. De daadwerkelijke z-index (van voor naar achter) van de weergaveobjecten is niet van belang, zolang de weergaveobjecten aan het weergaveoverzicht zijn toegevoegd. (Het maskerende object wordt alleen als masker op het scherm weergegeven.) Als het maskerende object een instantie MovieClip met meerdere frames is, worden alle frames in de tijdlijn precies zo afgespeeld als wanneer het object niet als masker zou fungeren. U kunt een masker verwijderen door de eigenschap `mask` in te stellen op `null`:

```
// remove the mask from mySprite
mySprite.mask = null;
```

U kunt een masker niet gebruiken om een ander masker te maskeren. U kunt de eigenschap `alpha` van een maskeerweergaveobject niet instellen. Er worden alleen vullingen gebruikt in een weergaveobject dat wordt gebruikt als een masker; lijnen worden genegeerd.

AIR 2

Als een gemaskeerd weergaveobject in de cache wordt opgenomen door het instellen van de eigenschappen `cacheAsBitmap` en `cacheAsBitmapMatrix`, moet het masker een onderliggend masker van het gemaskeerde weergaveobject zijn. En als het gemaskeerde weergaveobject afstamt van een weergaveobjectcontainer in de cache, moeten zowel het masker als het weergaveobject afstammen van de desbetreffende container. Als het gemaskeerde object afstamt van meerdere in de cache opgenomen weergaveobjectcontainers, moet het masker afstammen van de container in de cache die het meest op het gemaskeerde object in de weergavelijst lijkt.

Apparaatlettertypen maskeren

Met een weergaveobject kunt u tekst maskeren die wordt weergegeven in een apparaatlettertype. Wanneer u een weergaveobject gebruikt om tekst in een apparaatlettertype te maskeren, wordt het rechthoekige selectiekader van het masker gebruikt als maskeringsvorm. Wanneer u een niet-rechthoekig weergaveobjectmasker maakt voor tekst in een apparaatlettertype, heeft het masker dat wordt weergegeven in het SWF-bestand de vorm van het rechthoekig selectiekader van het masker, niet de vorm van het masker zelf.

Alpha-kanaalmaskering

Alpha-kanaalmaskering wordt ondersteund wanneer zowel voor het masker als voor de gemaskerde weergaveobjecten bitmapcaching wordt gebruikt, zoals hier aangegeven:

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

U kunt alpha-kanaalmaskering bijvoorbeeld gebruiken voor het toepassen van een filter op het maskerende object, onafhankelijk van een filter dat op het gemaskeerde weergaveobject is toegepast.

In het volgende voorbeeld wordt een extern afbeeldingsbestand in het werkgebied geladen. Deze afbeelding (of meer specifiek, de instantie Loader die hierin is geladen) wordt toegepast als het gemaskeerde weergaveobject. Een verlopende ovaal (effen zwart centrum dat transparant fadet naar de randen) wordt over de afbeelding getekend; dit is het alpha-masker. Voor beide weergaveobjecten is bitmapcaching ingeschakeld. De ovaal wordt als een masker voor de afbeelding ingesteld en wordt vervolgens sleepbaar gemaakt.

```
// 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);
```

Animaties van objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Animatie is een proces waardoor een object beweegt of in de loop van de tijd verandert. Gescripte animatie is een fundamenteel deel van videogames en wordt vaak gebruikt om Pools en bruikbare interactietips aan andere toepassingen toe te voegen.

De gedachte achter gescripte animaties is dat er een verandering moet plaatsvinden en dat die verandering moet worden verdeeld in toenames die worden uitgevoerd na een bepaalde tijd. U kunt in ActionScript met een algemene lusinstructie eenvoudig iets herhalen. Een lus doorloopt echter alle herhalingen voordat het scherm wordt bijgewerkt. Als u gescripte animatie wilt maken, moet u ActionScript schrijven die een actie herhaaldelijk uitvoert en ook het scherm bijwerkt wanneer de actie wordt uitgevoerd.

Stel dat u een eenvoudige animatie wilt maken, zoals een bal die over het scherm rolt. ActionScript bevat een simpele methode waarmee u het verloop van de tijd kunt volgen en het scherm overeenkomstig kunt bijwerken: u kunt dus code schrijven waardoor de bal telkens een klein stukje beweegt totdat de bestemming is bereikt. Na elke beweging wordt het scherm bijgewerkt, zodat de beweging over het werkgebied voor de gebruiker zichtbaar is.

Vanuit een praktisch standpunt is het nuttig om gescripte animatie en de framesnelheid van het SWF-bestand te synchroniseren (met andere woorden één animatieverandering uitvoeren wanneer een nieuw frame wordt weergegeven), omdat dit bepaalt hoe vaak Flash Player of AIR het scherm bijwerkt. Elk weergaveobject heeft een gebeurtenis `enterFrame` die op basis van de framesnelheid van het SWF-bestand wordt verzonden (één gebeurtenis per frame). De meeste ontwikkelaars die met gescripte animaties werken, gebruiken de gebeurtenis `enterFrame` om acties te maken die zich na verloop van tijd herhalen. U kunt code schrijven die naar de gebeurtenis `enterFrame` luistert, zodat de geanimeerde bal bij elk frame met een bepaalde afstand wordt verplaatst. Omdat het scherm wordt bijgewerkt (elk frame), wordt de bal op de nieuwe positie opnieuw getekend, waardoor er beweging ontstaat.

Opmerking: *Een andere methode waarmee u een actie in de loop van de tijd herhaaldelijk kunt uitvoeren is met gebruik van de klasse Timer. Een instantie Timer activeert een gebeurtenismelding op het moment dat een bepaalde hoeveelheid tijd is verstreken. U kunt code schrijven die animatie uitvoert door de timergebeurtenis van de klasse Timer te verwerken en de tijdsinterval op een kleine hoeveelheid in te stellen (een fractie van een seconde). Zie "Tijdintervallen bepalen" op pagina 4 voor meer informatie over het gebruik van de klasse Timer.*

In het volgende voorbeeld wordt er een cirkelinstantie Sprite met de naam `circle` in het werkgebied gemaakt. Wanneer de gebruiker op de cirkel klikt, wordt een gescripte animatie gestart, waardoor de `circle` gaat faden (zijn eigenschap `alpha` wordt verminderd) totdat hij volledig transparant is:

```
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);
```

Wanneer de gebruiker op de cirkel klikt, wordt de functie `fadeCircle()` als een listener op de gebeurtenis `enterFrame` geabonneerd, waardoor de functie vanaf nu één keer per frame wordt aangeroepen. Met deze functie wordt de `circle` gefadet doordat de eigenschap `alpha` wordt gewijzigd. Eenmaal per frame wordt de `alpha` van de cirkel met 0,05 (5 procent) verminderd, waarna het scherm wordt bijgewerkt. Wanneer de waarde van `alpha` is afgenomen tot 0 (de `circle` is volledig transparant), wordt de functie `fadeCircle()` verwijderd als een gebeurtenislistener, waardoor de animatie stopt.

U kunt dezelfde code gebruiken om bijvoorbeeld een animatiebeweging te maken in plaats van faden. Als u `alpha` in de gebeurtenislistenerfunctie `enterFrame` vervangt door een andere eigenschap, wordt die eigenschap geanimeerd. Als u bijvoorbeeld deze regel wijzigt

```
    circle.alpha -= .05;
```

naar deze code,

```
    circle.x += 5;
```

wordt de eigenschap `x` geanimeerd en wordt de cirkel naar rechts over het werkgebied bewogen. De voorwaarde waardoor de beweging stopt, kan worden gewijzigd om de beweging te laten stoppen (door het abonnement van de listener `enterFrame` op te zeggen) wanneer de gewenste x-coördinaat wordt bereikt.

Oriëntatie werkgebied

AIR 2.0 en hoger

De meeste mobiele apparaten veranderen de oriëntatie van de gebruikersinterface zodat de weergave rechtop blijft als de gebruiker het apparaat draait. Als u automatische oriëntatie inschakelt in uw toepassing, zorgt het apparaat voor de juiste stand van de weergave. U dient er echter voor te zorgen dat de inhoud er goed uitziet wanneer de hoogte-breedteverhouding van het werkgebied wordt gewijzigd. Als u automatische oriëntatie uitschakelt, staat de weergave van het apparaat vast tenzij u de oriëntatie handmatig wijzigt.

AIR-toepassingen kunnen op een aantal verschillende mobiele apparaten en besturingssystemen worden uitgevoerd. Het onderliggende oriëntatiegedrag kan per besturingssysteem en zelfs per apparaat op hetzelfde besturingssysteem variëren. De volgende eenvoudige ontwerpstrategie is een goede oplossing voor alle apparaten en besturingssystemen: schakel automatische oriëntatie in en luister naar Stage-gebeurtenissen `resize` om te bepalen of u de lay-out van de toepassing moet vernieuwen.

Als uw toepassing alleen ondersteuning biedt voor de verhouding Staand of de verhouding Liggend, kunt u automatische oriëntatie ook uitschakelen en de ondersteunde verhouding instellen in de AIR-toepassingsbeschrijving. Deze ontwerpstrategie leidt tot consistentie en kiest de "beste" oriëntatie voor de geselecteerde hoogte-breedteverhouding. Als u bijvoorbeeld de hoogte-breedteverhouding Liggend opgeeft, is de gekozen oriëntatie geschikt voor apparaten met uitschuifbare toetsenborden in de stand Liggend.

De actieve Stage-oriëntatie en hoogte-breedteverhouding opvragen

De oriëntatie wordt vermeld ten opzichte van de standaardpositie van het apparaat. Op de meeste apparaten is het duidelijke welke positie 'rechtop' is. Deze positie wordt als de *standaard*oriëntatie beschouwd. Er zijn drie andere mogelijke oriëntaties: *linksom gedraaid*, *rechtsom gedraaid* en *ondersteboven*. De klasse StageOrientation definieert de tekenreeksconstanten die moeten worden gebruikt voor het instellen of vergelijken van oriëntatiewaarden.

De Stage-klasse definieert twee eigenschappen die de oriëntatie vermelden:

- Stage.deviceOrientation — meldt de fysieke oriëntatie van het apparaat ten opzichte van de standaardpositie.

  Opmerking: *deviceOrientation is niet altijd beschikbaar wanneer de toepassing wordt gestart of wanneer het apparaat plat ligt. In deze gevallen wordt de apparaatoriëntatieonbekend gemeld.*

- Stage.orientation — meldt de oriëntatie van het werkgebied ten opzichte van de standaardpositie. Wanneer automatische oriëntatie ingeschakeld is, wordt het werkgebied in tegenovergestelde richting als het apparaat gedraaid om rechtop te blijven. De posities voor links en rechts die door de eigenschap `orientation` worden gemeld, zijn dus tegenovergesteld aan de posities die door de eigenschap `deviceOrientation` worden gemeld. Wanneer `deviceRotation` bijvoorbeeld *rotated right* meldt, meldt `orientation` juist *rotated left*.

De breedte-hoogteverhouding van het werkgebied kan worden afgeleid door eenvoudig de huidige breedte en hoogte van het werkgebied te vergelijken:

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ?
StageAspectRatio.LANDSCAPE : StageAspectRatio.PORTRAIT;
```

Automatische oriëntatie

Wanneer automatische oriëntatie is ingeschakeld en een gebruiker het apparaat roteert, verandert het besturingssysteem de oriëntatie van de volledige interface, inclusief de taakbalk en uw toepassing. Het resultaat is dat de hoogte-breedteverhouding van het werkgebied verandert van Staand in Liggend of andersom. Wanneer de hoogte-breedteverhouding verandert, veranderen de afmetingen van het werkgebied ook.

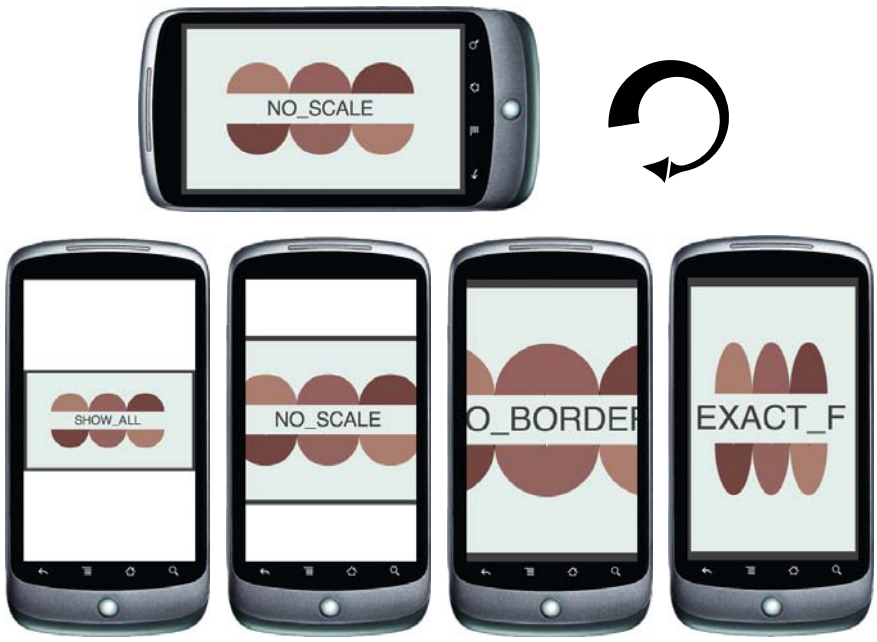
U schakelt automatische oriëntatie tijdens de runtime in of uit door de Stage-eigenschap `autoOrients` in te stellen op `true` of `false`. U kunt de aanvankelijke waarde van deze eigenschap instellen in de AIR-toepassingsbeschrijving met het element `<autoOrients>`. (Vóór AIR 2.6 is `autoOrients` een eigenschap met het kenmerk Alleen lezen die alleen kan worden ingesteld in de toepassingsbeschrijving.)

Als u een liggende of staande hoogte-breedteverhouding opgeeft en daarbij de optie voor automatische oriëntatie inschakelt, wordt deze optie in AIR beperkt tot de opgegeven hoogte-breedteverhouding.

Wijzigingen in de afmetingen van het werkgebied

Wanneer de afmetingen van het werkgebied veranderen, wordt de inhoud van het werkgebied geschaald en verplaatst volgens de eigenschappen `scaleMode` en `align` van het Stage-object. In de meeste gevallen kunt u beter niet vertrouwen op de automatische functionaliteit van de Stage-instellingen`scaleMode`, omdat dat vaak tot ongewenste resultaten leidt. U kunt beter de lay-out van uw afbeeldingen en componenten wijzigen of deze opnieuw tekenen om meerdere hoogte-breedteverhoudingen te ondersteunen. (Met een flexibele lay-outlogica werkt uw toepassing bovendien beter op verschillende apparaten met verschillende schermgrootten en breedte-hoogteverhoudingen.)

De volgende illustratie demonstreert de effecten van de verschillende `scaleMode`-instellingen bij het roteren van een standaard mobiel apparaat:



Rotatie van de hoogte-breedteverhouding Liggend naar Staand

De illustratie toont de schaling die optreedt tijdens het roteren van de hoogte-breedteverhouding Liggend naar de verhouding Staand in verschillende schaalmodi. Het roteren van Staand naar Liggend veroorzaakt vergelijkbare effecten.

Gebeurtenissen voor oriëntatiewijzigingen

Het Stage-object verzendt twee typen gebeurtenissen waarmee u wijzigingen in de oriëntatie kunt detecteren en erop kunt reageren. Er worden zowel `resize`- als `orientationChange`-gebeurtenissen voor het werkgebied verzonden wanneer automatische oriëntatie ingeschakeld is.

U kunt het beste de *resize*-gebeurtenis gebruiken als u automatische oriëntatie gebruikt om de weergave rechtop te houden. Wanneer het werkgebied een `resize`-gebeurtenis verzendt, past u de lay-out aan of tekent u de inhoud opnieuw, al naar gelang wat nodig is. De `resize`-gebeurtenis wordt alleen verzonden wanneer de modus voor het schalen van het werkgebied is ingesteld op `noScale`.

Met de `orientationChange`-gebeurtenis kunnen oriëntatiewijzigingen worden gedetecteerd. De `orientationChange`-gebeurtenis wordt alleen verzonden wanneer automatische oriëntatie is ingeschakeld.

Opmerking: *Op sommige mobiele platformen verzendt het werkgebied een annuleerbare `orientationChanging`-gebeurtenis voordat de gebeurtenissen resize of orientationChange worden verzonden. Aangezien deze gebeurtenis echter niet op elk platform wordt ondersteund, kunt u daar beter niet op vertrouwen.*

Handmatige oriëntatie

AIR 2.6 en hoger

U regelt de oriëntatie van het werkgebied met de Stage-methode `setOrientation()` of `setAspectRatio()`.

De oriëntatie van het werkgebied instellen

U kunt de oriëntatie van het werkgebied tijdens de runtime instellen met de methode `setOrientation()` van het Stage-object. Gebruik de tekenreeksconstanten die zijn gedefinieerd door de klasse StageOrientation om de gewenste oriëntatie op te geven:

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

Niet elk apparaat en besturingssysteem ondersteunt elke mogelijke oriëntatie. Zo biedt Android 2.2 bijvoorbeeld via de programmacode geen ondersteuning voor het kiezen van de links geroteerde oriëntatie op Staand/Liggende apparaten en wordt het ondersteboven draaien helemaal niet ondersteund. De eigenschap `supportedOrientations` van het werkgebied verschaft een lijst met oriëntaties die kunnen worden doorgegeven aan de methode `setOrientation()`:

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

De hoogte-breedteverhouding van het werkgebied instellen

Als de hoogte-breedteverhouding van het werkgebied voor u het belangrijkst is, kunt u deze verhouding instellen op Staand of Liggend. U kunt de breedte-hoogteverhouding instellen in het AIR-toepassingsbeschrijvingsbestand of tijdens de runtime met de Stage-methode `setAspectRatio()`:

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

De runtime kiest een van de twee mogelijke oriëntaties voor de opgegeven breedte-hoogteverhouding. Deze komt wellicht niet overeen met de huidige oriëntatie van het apparaat. Zo wordt bijvoorbeeld de standaardoriëntatie gekozen in plaats van ondersteboven (AIR 3.2 en lager), en de oriëntatie die geschikt is voor het uitschuifbare toetsenbord in plaats van de tegenovergestelde oriëntatie.

(AIR 3.3 en hoger) Vanaf AIR 3.3 (SWF-versie 16) kunt u ook de constante `StageAspectRatio.ANY` gebruiken. Als `Stage.autoOrients` is ingesteld op `true` en u `setAspectRatio(StageAspectRatio.ANY)` aanroept, kan uw toepassing zich aan alle oriëntaties aanpassen (Liggend-links, Liggend-rechts, Staand en Staand-ondersteboven). Nieuwe kenmerken van AIR 3.3 zijn de permanente hoogte-breedteverhouding en het feit dat extra rotaties van het apparaat beperkt worden tot de opgegeven oriëntatie.

Voorbeeld: de oriëntatie van het werkgebied afstemmen op de oriëntatie van het apparaat

Het volgende voorbeeld illustreert een functie die de oriëntatie van het werkgebied aanpast aan de huidige oriëntatie van het apparaat. De stage-eigenschap `deviceOrientation` geeft de fysieke oriëntatie van het apparaat aan, ook als de automatische oriëntatie is uitgeschakeld.

```
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
    }
}
```

De oriëntatie wordt asynchroon gewijzigd. U kunt naar de door het werkgebied verzonden gebeurtenis `orientationChange` luisteren om te zien of de wijziging is uitgevoerd. Als een oriëntatie niet wordt ondersteund op een apparaat, mislukt de aanroep `setOrientation()` zonder dat een fout wordt gemeld.

Scherminhoud dynamisch laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de volgende externe weergave-elementen in een ActionScript 3.0-toepassing laden:

- Een in ActionScript 3.0 gemaakt SWF-bestand. Dit kan een Sprite-, MovieClip- of andere klasse zijn waardoor Sprite wordt uitgebreid. In AIR-toepassingen op iOS kunnen alleen SWF-bestanden zonder ActionScript-bytecode worden geladen. Dit betekent dat SWF-bestanden met ingesloten gegevens, zoals afbeeldingen en geluid, wel kunnen worden geladen, maar SWF-bestanden met uitvoerbare code niet.
- Een afbeeldingsbestand, bijvoorbeeld JPG-, PNG- en GIF-bestanden.
- Een AVM1-SWF-bestand. Dit is een SWF-bestand dat in ActionScript 1.0 of 2.0 is geschreven. (niet ondersteund in mobiele AIR-toepassingen)

U laadt deze elementen met de klasse Loader.

Weergaveobjecten laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met een object Loader kunt u SWF-bestanden en bestanden met afbeeldingen in een toepassing laden. De klasse Loader is een subklasse van de klasse DisplayObjectContainer. Een object Loader kan slechts één onderliggend weergaveobject in het weergaveoverzicht bevatten, namelijk het weergaveobject dat het SWF-bestand of afbeeldingsbestand vertegenwoordigt dat door het object wordt geladen. Wanneer u een object Loader aan het weergaveoverzicht toevoegt, zoals in de volgende code, voegt u ook het geladen onderliggende weergaveobject aan het weergaveoverzicht toe zodra het is geladen:

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

Wanneer het SWF-bestand of de afbeelding is geladen, kunt u het geladen weergaveobject naar een andere weergaveobjectcontainer verplaatsen, zoals het DisplayObjectContainer-object `container` in dit voorbeeld:

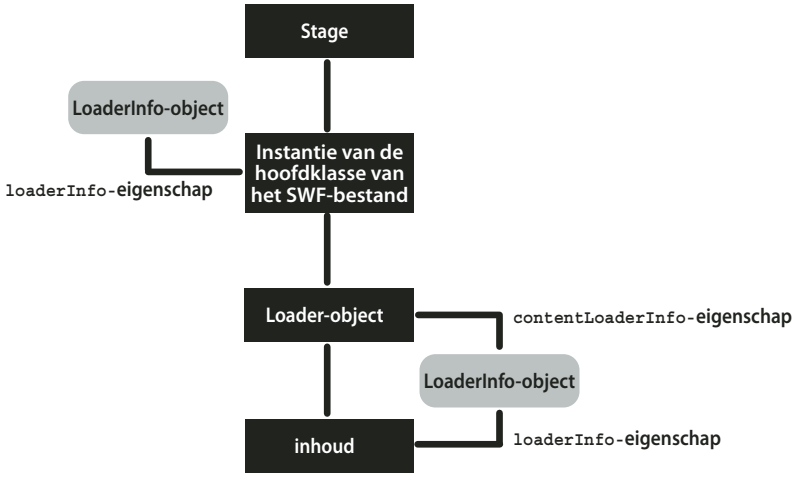
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

De voortgang van het laden controleren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer het bestand met laden begint, wordt er een object LoaderInfo gemaakt. Een object LoaderInfo biedt bijvoorbeeld informatie over de voortgang van het laden, de URL's van de lader en geladen inhoud, het totale aantal bytes voor de media en de nominale hoogte en breedte van de media. Een object LoaderInfo verzendt ook gebeurtenissen voor het controleren van de voortgang van het laden.

Het volgende diagram toont de verschillende manieren waarop het object LoaderInfo kan worden gebruikt (voor de instantie van de hoofdklasse van het SWF-bestand, voor een object Loader en voor een object dat is geladen door het object Loader):



U hebt toegang tot het object LoaderInfo als een eigenschap van zowel het object Loader en het geladen weergaveobject. Wanneer het laden begint, hebt u toegang tot het object LoaderInfo via de eigenschap `contentLoaderInfo` van het object Loader. Zodra het laden van het weergaveobject is voltooid, hebt u via de eigenschap `loaderInfo` van het weergaveobject ook toegang tot het object LoaderInfo als een eigenschap van het geladen weergaveobject. De eigenschap `loaderInfo` van het geladen weergaveobject verwijst naar hetzelfde object LoaderInfo als de eigenschap `contentLoaderInfo` van het object Loader. Een object LoaderInfo wordt dus gedeeld tussen een geladen object en het object Loader dat het object heeft geladen (tussen lader en de geladen inhoud).

Als u toegang wilt tot de eigenschappen van de geladen inhoud, moet u een gebeurtenislistener toevoegen aan het object LoaderInfo, zoals getoond in de volgende code:

```
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;
}
```

Zie “Gebeurtenissen afhandelen” op pagina 133 voor meer informatie.

De context van de geladen gegevens opgeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u via de methode `load()` of `loadBytes()` van de klasse Loader een extern bestand in Flash Player of AIR laadt, kunt u eventueel een parameter `context` opgeven. Deze parameter is een object LoaderContext.

De klasse LoaderContext bevat drie eigenschappen waarmee u kunt definiëren hoe de geladen inhoud wordt gebruikt:

- `checkPolicyFile`: Gebruik deze eigenschap alleen wanneer u een afbeeldingsbestand laadt (geen SWF-bestand). Als u deze eigenschap instelt op `true`, controleert de Loader de oorspronkelijke server op een beleidsbestand (zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116). Dit is alleen nodig voor inhoud afkomstig van een ander domein dan het domein van het SWF-bestand dat het object Loader bevat. Als de server bevoegdheden toekent aan het domein van Loader, heeft ActionScript in SWF-bestanden in het domein Loader toegang tot de gegevens in de geladen afbeelding. Met andere woorden, u kunt de opdracht `BitmapData.draw()` gebruiken om toegang te krijgen tot gegevens in de geladen afbeelding.

  Een SWF-bestand uit een ander domein dan het domein van het object Loader kan `Security.allowDomain()` aanroepen om een bepaald domein toe te staan.

- `securityDomain`: Gebruik deze eigenschap uitsluitend wanneer u een SWF-bestand laadt (geen afbeelding). Geef dit op voor een SWF-bestand uit een ander domein dan het domein van het bestand dat het object Loader bevat. Als u deze optie opgeeft, controleert Flash of er een beleidsbestand aanwezig is. Als dit er is, kunnen SWF-bestanden uit domeinen die in het domeinoverschrijdende beleidsbestand zijn toegestaan cross-scripting uitvoeren op de geladen SWF-inhoud. U kunt `flash.system.SecurityDomain.currentDomain` als deze parameter opgeven.

- `applicationDomain`: Gebruik deze eigenschap uitsluitend wanneer u een SWF-bestand laadt dat in ActionScript 3.0 is geschreven (geen afbeelding of SWF-bestand dat in ActionScript 1.0 of 2.0 is geschreven). Wanneer het bestand wordt geladen, kunt u opgeven dat het bestand in hetzelfde toepassingsdomein moet worden opgenomen als het domein van het object Loader. Dit doet u door de parameter `applicationDomain` in te stellen op `flash.system.ApplicationDomain.currentDomain`. Als u het geladen SWF-bestand in hetzelfde toepassingsdomein plaatst, hebt u direct toegang tot de klassen van het bestand. Dit kan nuttig zijn als u een SWF-bestand met ingesloten media laadt, waartoe u toegang hebt via de gekoppelde klassennamen. Zie "Werken met toepassingsdomeinen" op pagina 157 voor meer informatie.

In het volgende voorbeeld ziet u hoe u naar een beleidsbestand zoekt tijdens het laden van een bitmap uit een ander domein:

```
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);
```

In het volgende voorbeeld ziet u hoe u naar een beleidsbestand zoekt tijdens het laden van een SWF-bestand uit een ander domein, zodat u het bestand in dezelfde beveiligingssandbox als het Loader-object kunt plaatsen. Bovendien voegt de code de klassen in het geladen SWF-bestand toe aan hetzelfde toepassingsdomein als het Loader-object:

```
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);
```

Zie de LoaderContext-klasse in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

SWF-bestanden laden in AIR voor iOS

Adobe AIR 3.6 en later, alleen iOS

Op iOS-apparaten zijn er beperkingen op het laden en compileren van code bij uitvoering. Door deze beperkingen zijn er enkele noodzakelijke verschillen in de taak van het laden van externe SWF-bestanden naar uw toepassing:

- Alle SWF-bestanden die ActionScript-code bevatten, moeten worden opgenomen in het toepassingspakket. Geen SWF-bestand met code kan worden geladen vanaf een externe bron, zoals via een netwerk. Als onderdeel van het in een pakket plaatsen van de toepassing, wordt alle ActionScript-code in alle SWF-bestanden in het toepassingspakket gecompileerd naar native code voor iOS-apparaten.
- U kunt een SWF-bestand niet laden, verwijderen en vervolgens opnieuw laden. Als u dit probeert te doen, treedt er een fout op.
- Het gedrag van het laden naar het geheugen en het vervolgens verwijderen is hetzelfde als bij bureaubladplatforms. Als u een SWF-bestand laadt en het vervolgens verwijdert, worden alle visuele elementen in het SWF-bestand uit het geheugen verwijderd. Eventuele klasseverwijzingen naar een ActionScript-klasse in het geladen SWF-bestand echter blijven in het geheugen en zijn toegankelijk in ActionScript-code.
- Alle geladen SWF-bestanden moeten in hetzelfde toepassingsdomein als het SWF-hoofdbestand worden geladen. Dit is niet de standaardgedraging, dus voor elk SWF-bestand dat u laadt, moet u een LoaderContext-object maken dat het hoofdtoepassingsdomein opgeeft en dat LoaderContext-object doorsturen naar de Loader.load()-methodeaanroep. Als u een SWF-bestand probeert te laden in een ander toepassingsdomein dan het SWF-hoofdtoepassingsdomein, treedt er een fout op. Dit geldt zelfs als het geladen SWF-bestand alleen visuele elementen bevat en geen ActionScript-code.

  Het volgende voorbeeld toont de code die moet worden gebruikt om een SWF-bestand te laden vanaf een toepassingspakket naar het toepassingsdomein van het SWF-hoofdbestand:

```
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);
```

Een SWF-bestand laadt alleen elementen en er kan geen code worden geladen vanaf het toepassingspakket of via een netwerk. In elk geval moet het SWF-bestand nog steeds naar het hoofdtoepassingsdomein worden geladen.

Voor AIR-versies vóór AIR 3.6 wordt tijdens het compileren alle code van andere SWF-bestanden dan het SWF-bestand van de hoofdtoepassing verwijderd. SWF-bestanden die alleen visuele elementen bevatten, kunnen worden opgenomen in het toepassingspakket en bij uitvoering worden geladen, maar geen code. Als u een SWF-bestand dat ActionScript-code bevat, probeert te laden, treedt er een fout op. Door de fout verschijnt er een foutdialoogvenster "Niet-gecompileerde ActionScript" in de toepassing.

Zie ook

Meerdere SWF-bestanden in AIR-toepassingen op iOS in een pakket plaatsen en laden

De klassen ProLoader en ProLoaderInfo gebruiken

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger. Flash Professional CS5.5 is vereist.

Flash Professional CS5.5 introduceert de klassen fl.display.ProLoader en fl.display.ProLoaderInfo. Hiermee wordt het voorladen van een RSL nog eenvoudiger. Deze klassen vormen een spiegelbeeld van de klassen flash.display.Loader en flash.display.LoaderInfo, maar ze bieden een meer consistent laadgedrag.

Met de klasse ProLoader kunt u SWF-bestanden laden waarbij TLF wordt gebruikt in combinatie met het voorladen van een RSL. Tijdens runtime is bij SWF-bestanden die andere SWF-bestanden of SWZ-bestanden (zoals TLF) voorladen, een intern, omsluitend SWF-bestand vereist. Door deze extra complexiteit van het omsluitende SWF-bestand treedt soms ongewenst gedrag op. Dit probleem wordt opgelost door de ProLoader-klasse, waarmee dergelijke bestanden net zo worden geladen als standaard-SWF-bestanden. Deze oplossing is transparant voor de gebruiker en vereist geen special afhandeling in ActionScript. Bovendien wordt standaard-SWF-inhoud correct geladen door ProLoader.

In Flash Professional CS 5.5 en hoger kunt u veilig alle instanties van de Loader-klasse vervangen door instanties van de ProLoader-klasse. Vervolgens exporteert u uw toepassing naar Flash Player 10.2 of hoger, zodat de vereiste ActionScript-functionaliteit toegankelijk is voor ProLoader. U kunt ProLoader ook gebruiken wanneer u producten ontwerpt voor eerdere versies van Flash Player die ondersteuning bieden voor ActionScript 3.0. De volledige functionaliteit van ProLoader kan echter alleen worden ontsloten met Flash Player 10.2 of hoger. Gebruik altijd ProLoader wanneer u TLF in Flash Professional CS5.5 of hoger gebruikt. ProLoader is niet vereist in andere omgevingen dan Flash Professional.

Belangrijk: *Voor SWF-bestanden die zijn gepubliceerd in Flash Professional CS5.5 en hoger, kunt u altijd de klassen fl.display.ProLoader en fl.display.ProLoaderInfo gebruiken in plaats van de klassen flash.display.Loader en flash.display.LoaderInfo.*

Problemen die zijn opgelost door de ProLoader-klasse

De ProLoader-klasse biedt oplossingen voor problemen die niet kunnen worden afgehandeld door de bestaande Loader-klasse. De basis voor deze problemen ligt in het RSL-voorladen van TLF-bibliotheken. Dit is specifiek van toepassing op SWF-bestanden die een Loader-object gebruiken om andere SWF-bestanden te laden. De oplossing is onder meer van toepassing op de volgende problemen:

- **Scripting tussen het bestand dat wordt geladen en het al geladen bestand werkt niet zoals verwacht.** Door de ProLoader-klasse wordt het SWF-bestand dat wordt geladen, automatisch ingesteld als het bovenliggende element van het geladen SWF-bestand. Alle communicatie van het SWF-bestand dat wordt geladen wordt rechtstreeks verzonden naar het al geladen SWF-bestand.
- **De SWF-toepassing moet het laadproces actief beheren.** Hiertoe moeten extra gebeurtenissen worden geïmplementeerd, zoals `added`, `removed`, `addedToStage` en `removedFromStage`. Als uw toepassing is ontworpen voor Flash Player 10.2 of hoger, zijn deze extra stappen niet nodig dankzij ProLoader.

Code bijwerken van Loader naar ProLoader

Aangezien de ProLoader-klasse een spiegelbeeld is van de Loader-klasse, kunt u de oude Loader-klasse in uw code eenvoudig vervangen door de nieuwe Proloader-klasse. Het volgende voorbeeld laat zien hoe u de bestaande code kunt bijwerken naar de nieuwe klasse:

```
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!');
}
```

U kunt deze code als volgt bijwerken naar de ProLoader-klasse:

```
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!');
}
```

Voorbeeld van weergaveobjecten: SpriteArranger

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De voorbeeldtoepassing SpriteArranger borduurt verder op de voorbeeldtoepassing Geometric Shapes die afzonderlijk wordt beschreven in *ActionScript 3.0 leren gebruiken*.

De voorbeeldtoepassing SpriteArranger illustreert de beginselen voor het omgaan met weergaveobjecten:

- Weergaveobjectklassen uitbreiden
- Objecten toevoegen aan het weergaveoverzicht
- Lagen toekennen aan weergaveobjecten en werken met weergaveobjectcontainers
- Op weergaveobjectgebeurtenissen reageren
- De eigenschappen en methoden van weergaveobjecten gebruiken

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De SpriteArranger-toepassingsbestanden vindt u in de map Examples/SpriteArranger. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
SpriteArranger.mxml of SpriteArranger.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/SpriteArranger/CircleSprite.as	Een klasse die een objecttype Sprite definieert dat een cirkel op het scherm rendert.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	Een klasse die een canvas definieert. Dit is een weergaveobjectcontainer met objecten GeometricSprite.
com/example/programmingas3/SpriteArranger/SquareSprite.as	Een klasse die een objecttype Sprite definieert dat een vierkant op het scherm rendert.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	Een klasse die een objecttype Sprite definieert dat een driehoek op het scherm rendert.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	Een klasse die een object Sprite uitbreidt. De klasse wordt gebruikt om een vorm op het scherm te definiëren. De CircleSprite, SquareSprite en TriangleSprite breiden alledrie deze klasse uit.
com/example/programmingas3/geometricshapes/IGeometricShape.as	De basisinterface die de methoden definieert die door alle geometrische vormklassen moeten worden geïmplementeerd.
com/example/programmingas3/geometricshapes/IPolygon.as	Een interface die methoden definieert die moeten worden geïmplementeerd door geometrische vormklassen met meerdere zijden.
com/example/programmingas3/geometricshapes/RegularPolygon.as	Een type geometrische vorm met zijden van gelijke lengte die symmetrisch rond het midden van de vorm zijn geplaatst.
com/example/programmingas3/geometricshapes/Circle.as	Een geometrische vorm die een cirkel definieert.
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	Een subklasse van RegularPolygon die een driehoek met zijden van gelijke lengte definieert.
com/example/programmingas3/geometricshapes/Square.as	Een subklasse van RegularPolygon die een rechthoek met vier zijden van gelijke lengte definieert.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	Een klasse die een fabrieksmethode bevat voor het maken van vormen op basis van een opgegeven vormtype en grootte.

De SpriteArranger-klassen definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de toepassing SpriteArranger kan de gebruiker een aantal weergaveobjecten aan het canvas op het scherm toevoegen.

De klasse DrawingCanvas definieert een tekengebied, een type weergaveobjectcontainer, waarin een gebruiker op het scherm vormen kan toevoegen. Deze vormen op scherm zijn instanties van een van de subklassen van de klasse GeometricSprite.

De klasse DrawingCanvas

In Flex moeten alle onderliggende weergaveobjecten die aan een object Container worden toegevoegd, van een klasse zijn die van de klasse mx.core.UIComponent afstamt. Deze toepassing voegt een instantie van de klasse DrawingCanvas toe als een onderliggende instantie van een object mx.containers.VBox, zoals gedefinieerd in de MXML-code in het bestand SpriteArranger.mxml. Deze overerving wordt als volgt in de klassendeclaratie DrawingCanvas gedeclareerd:

```
public class DrawingCanvas extends UIComponent
```

De klasse UIComponent overerft van de klassen DisplayObject, DisplayObjectContainer en Sprite en de code in de klasse DrawingCanvas gebruikt de methoden en eigenschappen van die klassen.

De klasse DrawingCanvas breidt de klasse Sprite uit en deze overerving wordt als volgt gedefinieerd in de klassendeclaratie DrawingCanvas:

```
public class DrawingCanvas extends Sprite
```

De klasse Sprite is een subklasse van de klasse DisplayObjectContainer en DisplayObject en de klasse DrawingCanvas gebruikt de methoden en eigenschappen van deze klassen.

De constructormethode `DrawingCanvas()` stelt het Rectangle-object `bounds` in. Deze eigenschap wordt later gebruikt voor het tekenen van de contour van het canvas. Hierna roept de eigenschap de methode `initCanvas()` aan, als volgt:

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

Zoals getoond door het volgende voorbeeld, definieert de methode `initCanvas()` verschillende eigenschappen van het object DrawingCanvas, die als argumenten zijn doorgegeven aan de constructorfunctie:

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

De methode `initCanvas()` roept vervolgens de methode `drawBounds()` aan, die het canvas tekent met gebruik van de eigenschap `graphics` van de klasse DrawingCanvas. De eigenschap `graphics` is overgeërfd van de klasse 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();
```

De volgende extra methoden van de klasse DrawingCanvas worden aangeroepen op basis van gebruikersinteracties met de toepassing:

- De methoden `addShape()` en `describeChildren()`, die worden beschreven in "Weergaveobjecten aan het canvas toevoegen" op pagina 219
- De methoden `moveToBack()`, `moveDown()`, `moveToFront()` en `moveUp()`, die worden beschreven in "De weergaveobjectlagen opnieuw indelen" op pagina 222.
- De methode `onMouseUp()`, die wordt beschreven in "Weergaveobjecten klikken en slepen" op pagina 221

De klasse GeometricSprite en zijn subklassen

Elk weergaveobject dat de gebruiker aan het canvas kan toevoegen is een instantie van een van de volgende subklassen van de klasse GeometricSprite:

- CircleSprite
- SquareSprite
- TriangleSprite

De klasse GeometricSprite breidt de klasse flash.display.Sprite uit:

```
public class GeometricSprite extends Sprite
```

De klasse GeometricSprite bevat een aantal eigenschappen die worden gedeeld door alle objecten GeometricSprite. Deze eigenschappen worden in de constructorfunctie ingesteld op basis van de parameters die zijn doorgegeven aan de functie. Bijvoorbeeld:

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

De eigenschap `geometricShape` van de klasse GeometricSprite definieert een interface IGeometricShape, die de wiskundige eigenschappen maar niet de visuele eigenschappen van de vorm definieert. De klassen die de IGeometricShape-interface implementeren worden gedefinieerd in de voorbeeldtoepassing GeometricShapes die wordt beschreven in *ActionScript 3.0 leren gebruiken*.

De klasse GeometricSprite definieert de methode `drawShape()`, die nauwkeuriger wordt ingesteld in de overschrijfdefinities van elke subklasse van GeometricSprite. Zie voor meer informatie de sectie "Weergaveobjecten aan het canvas toevoegen" hieronder.

De klasse GeometricSprite biedt tevens de volgende methoden:

- De methoden `onMouseDown()` en `onMouseUp()`, die worden beschreven in "Weergaveobjecten klikken en slepen" op pagina 221
- De methoden `showSelected()` en `hideSelected()`, die worden beschreven in "Weergaveobjecten klikken en slepen" op pagina 221

Weergaveobjecten aan het canvas toevoegen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de gebruiker op de knop Vorm toevoegen klikt, roept de toepassing de methode `addShape()` van de klasse DrawingCanvas aan. Deze instantieert een nieuwe GeometricSprite door de toepasselijke constructorfunctie van een van de subklassen GeometricSprite aan te roepen, zoals getoond in het volgende voorbeeld:

```
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);
}
```

Elke constructormethode roept de methode `drawShape()` aan, die de eigenschap `graphics` van de klasse (die wordt overgeërfd van de klasse Sprite) gebruikt om de juiste vectorafbeelding te tekenen. De methode `drawShape()` van de klasse CircleSprite bevat bijvoorbeeld de volgende code:

```
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);
```

De tweede tot de laatste regel van de functie `addShape()` stellen de eigenschap `alpha` van het weergaveobject (die wordt overgeërfd van de klasse DisplayObject) in, zodat elk weergaveobject dat aan het canvas wordt toegevoegd, enigszins transparant is en de objecten hierachter zichtbaar voor de gebruiker zijn.

De laatste regel van de methode `addChild()` voegt het nieuwe weergaveobject aan de lijst met onderliggende items van de instantie van de klasse DrawingCanvas toe, die zich reeds in het weergaveoverzicht bevindt. Hierdoor wordt het nieuwe weergaveobject in het werkgebied weergegeven.

De interface voor de toepassing bevat de twee tekstvelden `selectedSpriteTxt` en `outputTxt`. De teksteigenschappen van deze tekstvelden worden bijgewerkt met de informatie over de objecten GeometricSprite die aan het canvas zijn toegevoegd of door de gebruiker zijn geselecteerd. De klasse GeometricSprite handelt deze informatierapporterende taak af door de methode `toString()` als volgt te overschrijven:

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

De eigenschap `shapeType` wordt ingesteld op de juiste eigenschap in de constructormethode van elke subklasse GeometricSprite. De methode `toString()` retourneert bijvoorbeeld de volgende waarde voor een instantie CircleSprite die recentelijk aan de instantie DrawingCanvas is toegevoegd:

```
Circle of size 50 at 0, 0
```

De methode `describeChildren()` van de klasse DrawingCanvas doorloopt de lijst met onderliggende items van het canvas met gebruik van de eigenschap `numChildren` (die is overgeërfd van de klasse DisplayObjectContainer) om de limiet van de lus `for` in te stellen. Er wordt als volgt een tekenreeks van elk onderliggend element gegenereerd:

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

De resulterende tekenreeks wordt gebruikt om de eigenschap `text` van het tekstveld `outputTxt` in te stellen.

Weergaveobjecten klikken en slepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer de gebruiker op een instantie GeometricSprite klikt, roept de toepassing de gebeurtenishandler `onMouseDown()` aan. Zoals getoond door de volgende coderegel, wordt de gebeurtenishandler ingesteld om naar gebeurtenissen muis omlaag in de constructorfunctie van de klasse GeometricSprite te luisteren:

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

De methode `onMouseDown()` roept vervolgens de methode `showSelected()` van het object GeometricSprite aan. Als het de eerste keer is dat deze methode voor het object wordt aangeroepen, maakt deze methode een nieuw object Shape met de naam `selectionIndicator` en gebruikt deze de eigenschap `graphics` van het object Shape om een rood gemarkeerde rechthoek te tekenen:

```
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);
```

Als de methode `onMouseDown()` al een keer is aangeroepen, stelt de methode alleen de eigenschap `visible` van `selectionIndicator` in (die wordt overgeërfd van de klasse DisplayObject):

```
this.selectionIndicator.visible = true;
```

De methode `hideSelected()` verbergt de vorm `selectionIndicator` van het eerder geselecteerde object door zijn eigenschap `visible` op `false` in te stellen.

De gebeurtenishandlermethode `onMouseDown()` roept ook de methode `startDrag()` aan (die is overgeërfd van de klasse Sprite), die de volgende code bevat:

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

Dit staat de gebruiker toe om het geselecteerde object in het canvas te slepen, binnen de grenzen die zijn ingesteld door de rechthoek `boundsRect`.

Wanneer de gebruiker de muisknop loslaat, wordt de gebeurtenis `mouseUp` verzonden. De constructormethode van DrawingCanvas stelt de volgende gebeurtenislistener in:

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

Deze gebeurtenislistener wordt ingesteld voor het object DrawingCanvas, in plaats van voor de individuele objecten GeometricSprite. Wanneer het object GeometricSprite wordt gesleept, kan dit object namelijk achter een ander weergaveobject eindigen (een ander object GeometricSprite) wanneer de muisknop wordt losgelaten. Het weergaveobject op de voorgrond ontvangt in dat geval de gebeurtenis muis omhoog en niet het weergaveobject dat door de gebruiker wordt gesleept. Als u de listener aan het object DrawingCanvas toevoegt, wordt de gebeurtenis altijd afgehandeld.

De methode `onMouseUp()` roept de methode `onMouseUp()` van het object GeometricSprite aan, die hierop de methode `stopDrag()` van het object GeometricSprite aanroept.

De weergaveobjectlagen opnieuw indelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De gebruikersinterface voor de toepassing bevat knoppen met het label Move Back (Naar achteren), Move Down (Omlaag), Move Up (Omhoog) en Move to Front (Naar voren). Wanneer de gebruiker op een van deze knoppen klikt, roept de toepassing de overeenkomende methode van de klasse DrawingCanvas aan: `moveToBack()`, `moveDown()`, `moveUp()` of `moveToFront()`. De methode `moveToBack()` bevat bijvoorbeeld de volgende code:

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

De methode gebruikt de methode `setChildIndex()` (die wordt overgeërfd van de klasse DisplayObjectContainer) om het weergaveobject op indexpositie 0 te plaatsen in de lijst met onderliggende items van de instantie DrawingCanvas (`this`).

De methode `moveDown()` werkt op ongeveer dezelfde manier, behalve dat deze de indexpositie van het weergaveobject met 1 vermindert in de lijst met onderliggende items van de instantie DrawingCanvas:

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

De methoden `moveUp()` en `moveToFront()` werken ongeveer op dezelfde manier als de methoden `moveToBack()` en `moveDown()`.

Hoofdstuk 11: Werken met geometrie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het pakket flash.geom bevat klassen die geometrische objecten definiëren, zoals punten, rechthoeken en transformatiematrices. U gebruikt deze klassen om de eigenschappen van objecten te definiëren die in andere klassen worden gebruikt.

Meer Help-onderwerpen

flash.geom, pakket

Basisbeginselen van geometrie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het pakket flash.geom bevat klassen die geometrische objecten definiëren, zoals punten, rechthoeken en transformatiematrices. Op zichzelf bieden deze klassen geen bijzondere functionaliteit, maar u gebruikt ze om de eigenschappen van objecten te definiëren die in andere klassen worden gebruikt.

Alle geometrieklassen zijn gebaseerd op de notie dat locaties op het scherm als een tweedimensionaal vlak worden weergegeven. Het scherm wordt gezien als een vlakke grafiek met een horizontale as (x) en een verticale as (y). Elke locatie (of *punt*) op het scherm kan worden aangeduid als een paar x- en y-waarden, ofwel de *coördinaten* van die locatie.

Elk weergaveobject, inclusief het werkgebied, heeft een eigen *coördinaatruimte*. De coördinaatruimte is een grafiek van het object voor het weergeven van de locaties van onderliggende weergaveobjecten, tekeningen enzovoorts. De *oorsprong* bevindt zich op coördinaatlocatie 0, 0 (waar de x- en de y-as elkaar kruisen) en wordt in de linkerbovenhoek van het weergaveobject geplaatst. De locatie van de oorsprong bevindt zich bij werkgebieden altijd op deze plaats, maar dit geldt niet per se voor andere weergaveobjecten. Hoe verder naar rechts op de x-as hoe hoger de waarden, hoe verder naar links hoe lager de waarden. Voor locaties die zich aan de linkerkant van de oorsprong bevinden, is de x-coördinaat negatief. In tegenstelling tot traditionele coördinatensystemen, worden in de Flash-runtime de coördinaatwaarden op y-as echter hoger in neerwaartse richting op het scherm en lager in opwaartse richting. De waarden boven de oorsprong hebben dus een negatieve y-coördinaat. Aangezien de linkerbovenhoek van het werkgebied de oorsprong van de coördinaatruimte is, hebben de meeste objecten in het werkgebied een x-coördinaat die groter is dan 0 en die kleiner is dan de breedte van het werkgebied. En de objecten hebben een y-coördinaat die groter is dan 0 en kleiner dan de hoogte van het werkgebied.

U kunt instanties van de klasse Point gebruiken om individuele punten in een coördinaatruimte te vertegenwoordigen. U kunt een Rectangle-instantie maken om een rechthoekig gebied in een coördinaatruimte te vertegenwoordigen. Geavanceerde gebruikers kunnen een Matrix-instantie gebruiken om meerdere of complexe transformaties toe te passen op een weergaveobject. Vele eenvoudige transformaties, zoals rotatie-, positie- en schaalwijzigingen, kunnen rechtstreeks op een object worden toegepast met behulp van de eigenschappen van dat object. Zie “Weergaveobjecten manipuleren” op pagina 184 voor meer informatie over het gebruik van eigenschappen van weergaveobjecten om transformaties toe te passen.

Belangrijke concepten en termen
De volgende referentielijst bevat belangrijke geometrietermen die in dit hoofdstuk worden gebruikt:

Cartesische coördinaten Coördinaten worden gewoonlijk genoteerd als een getallenpaar (zoals 5, 12 of 17, -23). De twee getallen zijn respectievelijk de x-coördinaat en de y-coördinaat.

Coördinaatruimte De grafiek van coördinaten vervat in een weergaveobject, waarop de onderliggende elementen van het weergaveobject zijn gepositioneerd.

Oorsprong Het punt in een coördinaatruimte waar de x-as en de y-as elkaar kruisen. Dit punt heeft de coördinaat 0, 0.

Punt Een enkele locatie in een coördinaatruimte. In het twee dimensionale coördinatensysteem dat in ActionScript wordt gebruikt, wordt het punt gedefinieerd door de locatie op x-as en de y-as (de coördinaten van het punt).

Registratiepunt De oorsprong (coördinaat 0, 0) van de coördinaatruimte van een weergaveobject.

Schalen De grootte van een object ten opzichte van de oorspronkelijke grootte ervan. Schalen als werkwoord betekent dat de grootte van een object wordt gewijzigd door het object uit te rekken of te verkleinen.

Transleren De coördinaten van een punt overzetten van de ene coördinaatruimte naar de andere.

Transformatie Een aanpassing in een visueel kenmerk van een afbeelding, zoals een object roteren, schalen, scheeftrekken of vervormen of de kleur ervan wijzigen.

x-as De horizontale as in het tweedimensionale coördinatensysteem dat in ActionScript wordt gebruikt.

y-as De verticale as in het tweedimensionale coördinatensysteem dat in ActionScript wordt gebruikt.

Point-objecten gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een Point-object definieert een Cartesisch-coördinatenpaar. Het vertegenwoordigt een locatie in een tweedimensionaal coördinatensysteem waarbij *x* de horizontale as vertegenwoordigt en *y* de verticale as.

U definieert een object Point door de eigenschappen `x` en `y` ervan als volgt te definiëren:

```
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;
```

De afstand tussen twee punten herleiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de methode `distance()` van de klasse Point gebruiken om de afstand tussen twee punten in een coördinaatruimte te herleiden. Met de volgende code wordt de afstand herleid tussen de registratiepunten van twee weergaveobjecten, `circle1` en `circle2`, in dezelfde weergaveobjectcontainer:

```
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);
```

Coördinaatruimten transleren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als twee weergaveobjecten zich in verschillende objectcontainers bevinden, kunnen ze zich in verschillende coördinaatruimten bevinden. U kunt de methode `localToGlobal()` van de klasse DisplayObject gebruiken om de coördinaten naar dezelfde (algemene) coördinaatruimte te transleren, die van het werkgebied. Met de volgende code wordt de afstand gevonden tussen de registratiepunten van twee weergaveobjecten, `circle1` en `circle2`, in de verschillende weergaveobjectcontainers:

```
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);
```

U kunt de methode `localToGlobal()` van de klasse DisplayObject ook gebruiken om erachter te komen wat de afstand is tussen het registratiepunt van een weergaveobject met de naam `target` en een specifiek punt in het werkgebied:

```
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);
```

Een weergaveobject verplaatsen met een opgegeven hoek en afstand

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de methode `polar()` van de klasse Point gebruiken om een weergaveobject een bepaalde afstand onder een bepaalde hoek te verplaatsen. Met de volgende code wordt bijvoorbeeld het object `myDisplayObject` met 100 pixels en een hoek van 60 graden verplaatst:

```
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;
```

Andere toepassingen van de klasse Point

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt Point-objecten gebruiken met de volgende methoden en eigenschappen:

Klasse	Methoden of eigenschappen	Beschrijving
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()getObjectsUnderPoint()`	Hiermee wordt een lijst met objecten onder een punt in een weergaveobjectcontainer geretourneerd.
BitmapData	`hitTest()`	Hiermee definieert u de pixel in het object BitmapData plus het punt dat u controleert als raakgebied.

BitmapData	`applyFilter()`
	`copyChannel()`
	`merge()`
	`paletteMap()`
	`pixelDissolve()`
	`threshold()`
		Hiermee definieert u de posities van rechthoeken die de bewerkingen definiëren.

Matrix	`deltaTransformPoint()` `transformPoint()`	Hiermee definieert u punten waarvoor u een transformatie wilt toepassen.

Rechthoek	`bottomRight`
	`size`
	`topLeft`
		Hiermee definieert u deze eigenschappen.

Rectangle-objecten gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een Rectangle-object definieert een rechthoekig gebied. Een Rectangle-object heeft een positie, gedefinieerd door de coördinaten *x* en *y* van de linkerbovenhoek ervan, een eigenschap `width` en een eigenschap `height`. U kunt deze eigenschappen definiëren voor een nieuw Rectangle-object door de constructorfunctie `Rectangle()` als volgt aan te roepen:

```
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);
```

De grootte en positie van Rectangle-objecten wijzigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn enkele manieren om de grootte en positie van Rectangle-objecten te wijzigen.

U kunt de positie van het object Rectangle rechtstreeks wijzigen door de eigenschappen `x` en `y` ervan te wijzigen. Deze wijziging is niet van invloed op de breedte of hoogte van het Rectangle-object.

```
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)
```

Als u de eigenschap `left` of de eigenschap `top` van een Rectangle-object wijzigt, wordt de rechthoek geherpositioneerd. Dit wordt door de volgende code gedemonstreerd. De eigenschappen `x` en `y` van de rechthoek komen overeen met respectievelijk de eigenschappen `left` en `top`. De positie van de linkerbenedenhoek van het Rectangle-object verandert echter niet, zodat de grootte wordt gewijzigd.

```
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)
```

Zo ziet u in het volgende voorbeeld dat bij het wijzigen van de eigenschap `bottom` of `right` van een Rectangle-object de positie van de linkerbovenhoek niet verandert. De rechthoek wordt dienovereenkomstig gewijzigd:

```
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)
```

U kunt de positie van een object Rectangle ook als volgt wijzigen met behulp van de methode `offset()`:

```
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)
```

De methode `offsetPt()` werkt op een soortgelijke manier. Alleen wordt daarbij een object Point als parameter gebruikt in plaats van verschuivingswaarden voor *x* en *y*.

U kunt ook de grootte van een object Rectangle wijzigen met de methode `inflate()` en de twee parameters `dx` en `dy`. De parameter `dx` representeert het aantal pixels waarmee de linker- en de rechterkant van de rechthoek van het midden worden verplaatst. De parameter `dy` representeert het aantal pixels waarmee de boven- en de onderkant van de rechthoek van het midden worden verplaatst:

```
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)
```

De methode `inflatePt()` werkt op een soortgelijke manier. Alleen wordt daarbij een object Point als parameter gebruikt in plaats van waarden voor `dx` en `dy`.

Samenvoegingen en snijpunten van Rectangle-objecten zoeken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `union()` kunt u het rechthoekig gebied vinden dat wordt gevormd door de randen van twee rechthoeken:

```
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)
```

Met de methode `intersection()` kunt u het rechthoekige gebied vinden dat wordt gevormd door de overlappende gebieden van twee rechthoeken:

```
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)
```

Met de methode `intersects()` kunt u nagaan of twee rechthoeken elkaar snijden. Met de methode `intersects()` kunt u ook nagaan of een weergaveobject zich in een bepaald gebied van het werkgebied bevindt. Ga er in de volgende codevoorbeeld van uit dat de coördinaatruimte van de weergaveobjectcontainer met het object `circle` gelijk is aan die van het werkgebied. Het voorbeeld laat zien hoe u met de methode `intersects()` kunt bepalen of een weergaveobject, `circle`, bepaalde gebieden in het werkgebied snijdt, gedefinieerd door de Rectangle-objecten `target1` en `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
```

Met de methode `intersects()` kunt u zo ook nagaan of de selectierechthoeken van twee weergaveobjecten elkaar overlappen. Gebruik de methode `getRect()` van de klasse DisplayObject om de extra ruimte op te geven die door de streken van een weergaveobject aan een selectiegebied wordt toegevoegd.

Andere toepassingen van het object Rectangle

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt Rectangle-objecten gebruiken in de volgende methoden en eigenschappen:

Klasse	Methoden of eigenschappen	Beschrijving
BitmapData	`applyFilter()`, `colorTransform()`, `copyChannel()`, `copyPixels()`, `draw()`, `drawWithQuality()`, `encode()`, `fillRect()`, `generateFilterRect()`, `getColorBoundsRect()`, `getPixels()`, `merge()`, `paletteMap()`, `pixelDissolve()`, `setPixels()` en `threshold()`	Hiermee geeft u het type op voor sommige parameters om een gebied van het object BitmapData te definiëren.
DisplayObject	`getBounds()`, `getRect()`, `scrollRect`, `scale9Grid`	Hiermee geeft u het gegevenstype op voor de eigenschap of het gegevenstype dat wordt geretourneerd.
PrintJob	`addPage()`	Hiermee definieert u de parameter `printArea`.
Sprite	`startDrag()`	Hiermee definieert u de parameter `bounds`.
TextField	`getCharBoundaries()`	Hiermee geeft u het type van de geretourneerde waarde op.
Transform	`pixelBounds`	Hiermee geeft u het gegevenstype op.

Matrix-objecten gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De Matrix-klasse representeert een transformatiematrix waarmee wordt bepaald hoe punten worden toegewezen van een coördinaatruimte naar een andere. U kunt verschillende grafische transformaties op een weergaveobject uitvoeren door de eigenschappen van een object Matrix in te stellen, dat object Matrix toe te passen op de eigenschap `matrix` van een object Transform en vervolgens dat object Transform toe te passen als de eigenschap `transform` van het weergaveobject. Deze transformatiefuncties omvatten transleren (positie van *x* en *y* wijzigen), roteren, schalen en scheeftrekken.

Hoewel u een matrix kunt definiëren door de eigenschappen (`a`, `b`, `c`, `d`, `tx`, `ty`) van een object Matrix rechtstreeks aan te passen, is het gebruik van de methode `createBox()` eenvoudiger. Deze methode maakt gebruik van parameters waarmee u de effecten voor schalen, roteren en transleren voor de resulterende matrix rechtstreeks definieert. Met de volgende code wordt bijvoorbeeld een Matrix-object gemaakt waarmee het object horizontaal met 2,0 wordt geschaald, verticaal met 3,0 wordt geschaald, met 45 graden wordt geroteerd, en met 10 pixels naar rechts en 20 pixels naar beneden wordt verplaatst (getransleerd):

```
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);
```

U kunt de effecten voor schalen, roteren en transleren van een object Matrix ook aanpassen met de methoden `scale()`, `rotate()` en `translate()`. Deze methoden worden met de waarden van het bestaande Matrix-object gecombineerd. Met de volgende code wordt bijvoorbeeld een Matrix-object ingesteld waarmee een object met een factor 4 wordt geschaald en 60 graden wordt geroteerd, aangezien de methoden `scale()` en `rotate()` tweemaal worden aangeroepen:

```
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;
```

Als u een object Matrix wilt scheeftrekken, past u de eigenschap `b` of `c` ervan aan. Bij aanpassing van de eigenschap `b` wordt de matrix verticaal scheefgetrokken en bij aanpassing van de eigenschap `c` horizontaal. Met de volgende code wordt het Matrix-object `myMatrix` met een factor 2 verticaal scheefgetrokken:

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

U kunt een matrixtransformatie toepassen op de eigenschap `transform` van een weergaveobject. Met de volgende code wordt bijvoorbeeld een matrixtransformatie toegepast op een weergaveobject met de naam `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;
```

Met de eerste regel wordt een object Matrix ingesteld op de bestaande transformatiematrix die door het weergaveobject `myDisplayObject` wordt gebruikt (de eigenschap `matrix` van de eigenschap `transformation` van het weergaveobject `myDisplayObject`). Op deze wijze hebben methoden van de klasse Matrix die u aanroept, een cumulatief effect op de huidige positie, schaal en rotatie van het weergaveobject.

Opmerking: *De klasse ColorTransform is ook onderdeel van het pakket flash.geometry. Met deze klasse stelt u de eigenschap* `colorTransform` *in van een object Transform. Omdat hierbij geen sprake is van een geometrische transformatie, wordt deze klasse niet in dit hoofdstuk besproken. Zie de ColorTransform-klassse in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.*

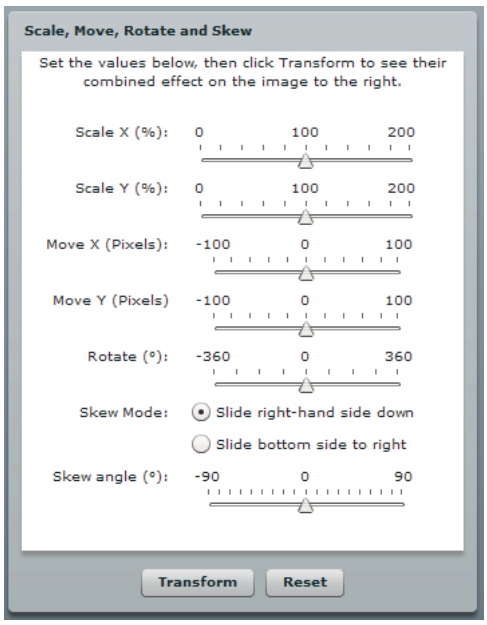
Voorbeeld van geometrie: een matrixtransformatie toepassen op een weergaveobject

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de voorbeeldtoepassing DisplayObjectTransformer worden enkele kenmerken van het gebruik van de klasse Matrix getoond om een weergaveobject te transformeren, zoals:

- Het weergaveobject roteren
- Het weergaveobject schalen
- Het weergaveobject transleren (verplaatsen)
- Het weergaveobject scheeftrekken

De toepassing biedt een interface voor het aanpassen van de parameters van de matrixtransformatie, als volgt:



Wanneer de gebruiker op de knop Transform klikt, wordt in de toepassing de desbetreffende transformatie toegepast.

Het oorspronkelijke weergaveobject en het weergaveobject -45° geroteerd en met 50% geschaald

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van DisplayObjectTransformer vindt u in de map Samples/DisplayObjectTransformer. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
DisplayObjectTransformer.mxml of DisplayObjectTransformer.fla	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML)
com/example/programmingas3/geometry/MatrixTransformer.as	Een klasse met methoden voor de toepassing van matrixtransformaties.
img/	Een map met voorbeeldafbeeldingsbestanden die in de toepassing worden gebruikt.

De klasse MatrixTransformer definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse MatrixTransformer bevat statische methoden voor de toepassing van geometrische transformaties op objecten Matrix.

De methode transform()

De methode `transform()` bevat parameters voor:

- `sourceMatrix` - De invoermatrix die wordt getransformeerd
- `xScale` en `yScale` - De *x*- en *y*-schaalfactor
- `dx` en `dy` - De mate waarin *x* en *y* worden getransleerd, in pixels
- `rotation` - De mate van rotatie, in graden
- `skew` - De scheeftrekfactor, als percentage
- `skewType` - De richting waarin wordt scheefgetrokken, `"right"` of `"left"`

De geretourneerde waarde is de resulterende matrix.

De methode `transform()` roept de volgende statische methoden aan van de klasse:

- `skew()`
- `scale()`
- `translate()`

- `rotate()`

Elke methode retourneert de bronmatrix met de toegepaste transformatie.

De methode skew()

De methode `skew()` trekt de matrix scheef door de eigenschappen `b` en `c` van de matrix aan te passen. Een optionele parameter, `unit`, bepaalt de gebruikte eenheden om de scheeftrekhoek te definiëren. De methode zet zo nodig de waarde voor `angle` om in radialen:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

Er wordt een object Matrix A met de naam `skewMatrix` gemaakt en aangepast om de scheeftrektransformatie toe te passen. Aanvankelijk is er sprake van de identiteitsmatrix, als volgt:

```
var skewMatrix:Matrix = new Matrix();
```

De parameter `skewSide` bepaalt de kant waarop het scheeftrekken wordt toegepast. Bij `"right"` wordt met de volgende code de eigenschap `b` van de matrix ingesteld:

```
skewMatrix.b = Math.tan(angle);
```

Anders wordt de onderkant scheefgetrokken door de eigenschap `c` van de matrix in te stellen, als volgt:

```
skewMatrix.c = Math.tan(angle);
```

Het resulterende scheeftrekken wordt vervolgens toegepast op de bestaande matrix door de twee matrices samen te voegen, zoals in het volgende voorbeeld wordt getoond:

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

De methode scale()

Zoals u in het volgende voorbeeld kunt zien, wordt met de methode `scale()` eerst de schaalfactor aangepast als deze als percentage wordt opgegeven. Vervolgens wordt de methode `scale()` van het matrixobject gebruikt:

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

De methode translate()

De methode `translate()` past eenvoudig de translatiefactoren `dx` en `dy` toe door de methode `translate()` van het matrixobject aan te roepen, als volgt:

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

De methode rotate()

De methode `rotate()` zet de invoerrotatiefactor om in radialen (als deze als graden of verloop wordt opgegeven). Vervolgens wordt de methode `rotate()` van het matrixobject aangeroepen:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

De methode MatrixTransformer.transform() vanuit de toepassing aanroepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing bevat een gebruikersinterface om de transformatieparameters van de gebruiker te verkrijgen. Deze waarden worden vervolgens doorgegeven, samen met de eigenschap `matrix` van de eigenschap `transform` van het weergaveobject, aan de methode `Matrix.transform()`. Dit gaat als volgt:

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

In de toepassing worden de geretourneerde waarden vervolgens toegepast op de eigenschap `matrix` van de eigenschap `transform` van het weergaveobject, waarbij de transformatie wordt geïnitieerd:

```
img.content.transform.matrix = tempMatrix;
```

Hoofdstuk 12: De teken-API gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Hoewel geïmporteerde afbeeldingen en illustraties belangrijk zijn, geeft de functionaliteit die we teken-API noemen en waarmee u lijnen en vormen in ActionScript kunt tekenen, u de vrijheid een toepassing te starten met de computerequivalent van een leeg canvas waarop u elke afbeelding kunt maken. U kunt zo uw eigen afbeeldingen maken waardoor u meer mogelijkheden voor uw toepassingen hebt. Met de technieken die hier werden beschreven, kunt u onder andere een tekenprogramma maken, geanimeerde, interactieve afbeeldingen maken of uw eigen gebruikersinterface-elementen programmatisch maken.

Meer Help-onderwerpen

flash.display.Graphics

Basisbeginselen van de teken-API

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De API voor tekenen is de naam voor de ActionScript-functionaliteit waarmee u vectorafbeeldingen kunt maken, waaronder lijnen, curven, vormen, opvullingen en verlopen. U kunt ze met ActionScript op het scherm weergeven. De klasse flash.display.Graphics biedt deze functionaliteit. U kunt met ActionScript op elke Shape-, Sprite- of MovieClip-instantie tekenen met de eigenschap `graphics`, die in elk van deze klassen is gedefinieerd. (Elke `graphics`-eigenschap van die klassen is eigenlijk een instantie van de klasse Graphics.)

Als u nog weinig ervaring hebt met tekenen met code, heeft de klasse Graphics enkele methoden waarmee u gemakkelijk veelvoorkomende vormen kunt tekenen, zoals cirkels, ellipsen, rechthoeken en rechthoeken met afgeronde hoeken. U kunt ze als lege lijnen of gevulde vormen tekenen. Wanneer u geavanceerde functionaliteit nodig hebt, heeft de klasse Graphics ook methoden voor het tekenen van lijnen en kwadratische Bézier-curven die u samen met de trigonometriefuncties van de klasse Math kunt gebruiken om een willekeurige vorm te tekenen.

Flash-runtimes (zoals Flash Player 10 en Adobe AIR 1.5 en latere versies) voegen een bijkomende API voor tekenen toe, waarmee u programmatisch volledige vormen kunt tekenen met behulp van één opdracht. Als u vertrouwd bent met de klasse Graphics en de taken die worden behandeld in 'Basisbeginselen van de teken-API', gaat u verder met “Geavanceerd gebruik van de teken-API” op pagina 248 om meer te leren over deze functies van de API voor tekenen.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het gebruik van de API voor tekenen:

Ankerpunt Een van de twee eindpunten van een kwadratische Bézier-curve.

Besturingspunt Het punt dat de richting en mate van de curve van een kwadratische Bézier-curve definieert. De gekromde lijn bereikt nooit het besturingspunt; de lijn kromt alsof deze naar het besturingspunt wordt getrokken.

Coördinaatruimte De grafiek van coördinaten vervat in een weergaveobject, waarop de onderliggende elementen van het weergaveobject zijn gepositioneerd.

Vulling Het effen binnenste deel van een vorm waarvan een lijn is opgevuld met kleur of een volledige vorm zonder omtrek.

Verloop Een kleur die bestaat uit een geleidelijke overgang van een kleur naar een of meer andere kleuren (in tegenstelling tot een effen kleur).

Punt Een enkele locatie in een coördinaatruimte. In het tweedimensionale coördinatensysteem dat in ActionScript wordt gebruikt, wordt een punt gedefinieerd door de locatie ervan langs de x-as en de y-as (de coördinaten van het punt).

Kwadratische Bézier-curve Een curvetype dat wordt gedefinieerd door een bepaalde wiskundige formule. Bij dit curvetype wordt de vorm van de curve berekend op basis van de positie van de ankerpunten (de eindpunten van de curve) en een besturingspunt dat de mate en richting van de curve definieert.

Schalen De grootte van een object ten opzichte van de oorspronkelijke grootte ervan. Schalen als werkwoord betekent dat de grootte van een object wordt gewijzigd door het object uit te rekken of te verkleinen.

Streek Het omtrekdeel van een vorm met een lijn die is opgevuld met kleur, of de lijnen van een niet-opgevulde vorm.

Transleren De coördinaten van een punt overzetten van de ene coördinaatruimte naar de andere.

x-as De horizontale as in het tweedimensionale coördinatensysteem dat in ActionScript wordt gebruikt.

y-as De verticale as in het tweedimensionale coördinatensysteem dat in ActionScript wordt gebruikt.

De klasse Graphics

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elk object Shape, Sprite en MovieClip heeft een eigenschap `graphics`; een instantie van de klasse Graphics. De klasse Graphics heeft eigenschappen en methoden voor het tekenen van lijnen, vullingen en vormen. Wanneer u een weergaveobject alleen wilt gebruiken als canvas voor het tekenen van inhoud, kunt u een instantie Shape gebruiken. Een instantie Shape werkt beter dan andere weergaveobjecten voor tekenen, omdat er geen overhead van de extra functionaliteit in de klassen Sprite en MovieClip is. Wanneer u een weergaveobject wilt waarop u grafische inhoud kunt tekenen en u wilt dat dat object andere weergaveobjecten bevat, kunt u een instantie Sprite gebruiken. Zie "Een subklasse DisplayObject kiezen" op pagina 183 voor meer informatie over hoe u bepaalt welk weergaveobject u gebruikt voor verschillende taken.

Lijnen en curven tekenen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Alles wat u met een instantie Graphics tekent, is gebaseerd op de basisfunctie tekenen met lijnen en curven. Daarom moet alles wat met de tekenfunctie van ActionScript wordt gedaan worden uitgevoerd met dezelfde reeks stappen:

- Definieer lijn- en vullingstijlen.
- Stel de oorspronkelijke tekenpositie in.
- Teken lijnen, curven en vormen (u kunt het tekenpunt eventueel verplaatsen).
- Eindig eventueel met een vulling.

Lijn- en vullingstijlen definiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u met de eigenschap `graphics` van een instantie Shape, Sprite of MovieClip wilt tekenen, moet u eerst de stijl (lijngrootte en kleur, vulkleur) definiëren die bij het tekenen wordt gebruikt. Net als wanneer u de tekengereedschappen in Adobe® Flash® Professional of andere tekentoepassingen gebruikt, kunt u in ActionScript met of zonder streek en met of zonder vulkleur tekenen. U geeft het uiterlijk van de streek op met de methode `lineStyle()` of `lineGradientStyle()`. Als u een effen lijn wilt maken, gebruikt u de methode `lineStyle()`. Wanneer u deze methode aanroept, zijn de eerste drie parameters die u opgeeft de meest voorkomende waarden: lijndikte, -kleur en -alfa. Deze coderegel instrueert de Shape met de naam `myShape` lijnen te tekenen die 2 pixels dik, rood (0x990000) en 75% dekkend zijn:

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

De standaardwaarde voor de parameter alpha is 1,0 (100%), dus u kunt deze parameter uitgeschakeld laten wanneer de lijn volledig dekkend moet zijn. De methode `lineStyle()` accepteert ook twee bijkomende parameters voor pixelhints en de schaalmodus. Zie de beschrijving van de methode `Graphics.lineStyle()` in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over het gebruik van deze parameters.

U kunt een verlooplijn maken met de methode `lineGradientStyle()`. Deze methode wordt beschreven in "Verlooplijnen en vullingen maken" op pagina 240.

Wanneer u een gevulde vorm wilt maken, roept u de methode `beginFill()`, `beginGradientFill()`, `beginBitmapFill()` of `beginShaderFill()` aan voordat u begint met tekenen. De eenvoudigste methode van deze drie, de methode `beginFill()`, accepteert twee parameters: de vulkleur en (optioneel) een alfawaarde voor de vulkleur. Wanneer u bijvoorbeeld een vorm met een effen groene vulling wilt maken, gebruikt u de volgende code (we gaan er vanuit dat er op een object met de naam `myShape` wordt getekend):

```
myShape.graphics.beginFill(0x00FF00);
```

Wanneer een vulmethode wordt aangeroepen, wordt een eerdere vulling impliciet beëindigd voordat er een nieuwe wordt gestart. Wanneer u een methode aanroept die een streekstijl opgeeft, wordt de vorige streek vervangen, maar wordt geen eerder opgegeven vulling gewijzigd en vice versa.

Wanneer u de lijnstijl en vuleigenschappen hebt opgegeven, is de volgende stap het beginpunt voor uw tekening aangeven. De instantie Graphics heeft een tekenpunt, net als de punt van een pen op een vel papier. Op de plek waar het tekenpunt staat, begint de volgende tekenhandeling. Een object Graphics begint met het tekenpunt op het punt 0, 0 in de coördinaatruimte van het object waarop wordt getekend. Wanneer u op een ander punt met tekenen wilt beginnen, roept u eerst de methode `moveTo()` aan voordat u een van de tekenmethoden aanroept. Dit is hetzelfde als de punt van een pen van het papier halen en deze op een andere plaats weer neerzetten.

Wanneer het tekenpunt op zijn plaats staat, tekent u met behulp van een reeks aanroepen van de tekenmethoden `lineTo()` (voor rechte lijnen) en `curveTo()` (voor gekromde lijnen).

Terwijl u tekent, kunt u de methode `moveTo()` altijd aanroepen om het tekenpunt op een nieuwe positie te zetten zonder te tekenen.

Als u een vulkleur hebt opgegeven terwijl u tekent, kunt u de vulling uitschakelen door de methode `endFill()` aan te roepen. Als u geen gesloten vorm hebt getekend (met andere woorden als het tekenpunt niet samenvalt met het beginpunt van de vorm op het moment dat u de methode `endFill()` aanroept) en u de methode `endFill()` aanroept, wordt de vorm automatisch gesloten door Flash-runtime door een rechte lijn te tekenen van het huidige tekenpunt naar de locatie die is opgegeven in de meest recente aanroep van `moveTo()`. Als u met een vulling bent begonnen en geen aanroep van `endFill()` hebt uitgevoerd, wordt de huidige vulling gesloten wanneer u `beginFill()` (of een van de andere vulmethoden) aanroept en wordt een nieuwe vulling gestart.

Rechte lijnen tekenen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de methode `lineTo()` aanroept, tekent het object Graphics een rechte lijn vanaf het huidige tekenpunt naar de coördinaten die u opgeeft als de twee parameters in de methodeaanroep, waarbij de lijnstijl wordt getekend die u hebt opgegeven. De volgende coderegel zet bijvoorbeeld het tekenpunt op het punt 100, 100 en tekent vervolgens een lijn naar het punt 200, 200:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

In het volgende voorbeeld worden rode en groene driehoeken met een hoogte van 100 pixels getekend:

```
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);
```

Curven tekenen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `curveTo()` tekent een kwadratische Bézier-curve. Er wordt een boog getekend die op twee punten (ankerpunten) samenkomt en naar een derde punt (het besturingspunt) toebuigt. Het object Graphics gebruikt de huidige tekenpositie als het eerste ankerpunt. Wanneer u de methode `curveTo()` aanroept, geeft u vier parameters door: de x- en y-coördinaten van het besturingspunt, gevolgd door de x- en y-coördinaten van het tweede ankerpunt. De volgende code tekent bijvoorbeeld een curve die begint bij punt 100, 100 en eindigt bij punt 200, 200. Omdat het besturingspunt zich op punt 175, 125 bevindt, wordt een curve gemaakt die eerst naar rechts en vervolgens omlaag loopt:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

In het volgende voorbeeld worden rood met groene ronde objecten met een breedte en hoogte van 100 pixels getekend. De cirkels zijn echter niet perfect door de aard van de kwadratische Bézier-vergelijking:

```
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);
```

Vormen met ingebouwde methoden tekenen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het tekenen van veelvoorkomende vormen zoals cirkels, ellipsen, rechthoeken en rechthoeken met afgeronde hoeken is vereenvoudigd, omdat ActionScript 3.0 methoden heeft die deze veelvoorkomende vormen voor u tekenen. Deze methoden van de klasse Graphics zijn de volgende: `drawCircle()`, `drawEllipse()`, `drawRect()` en `drawRoundRect()`. Deze methoden kunnen worden gebruikt in plaats van de methoden `lineTo()` en `curveTo()`. U moet echter wel de lijn- en vullingstijlen opgeven voordat u deze methoden aanroept.

In het volgende voorbeeld wordt het voorbeeld met rode, groene en blauwe vierkanten met een breedte en hoogte van 100 pixels, opnieuw gemaakt. De code gebruikt de methode `drawRect()` en geeft daarnaast op dat de vulkleur een alfa heeft van 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);
```

Bij een object Sprite of MovieClip wordt de tekeninhoud die met de eigenschap `graphics` is gemaakt, altijd achter alle onderliggende weergaveobjecten in het object weergegeven. De inhoud van de eigenschap `graphics` is ook geen apart weergaveobject, dus dit wordt niet in de lijst van de onderliggende items van het object Sprite of MovieClip weergegeven. Zo heeft het volgende object Sprite bijvoorbeeld een cirkel die met de eigenschap `graphics` is getekend en heeft het een object TextField in de lijst met onderliggende weergaveobjecten:

```
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);
```

TextField wordt vóór de cirkel weergegeven die met het object graphics is getekend.

Verlooplijnen en vullingen maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het object graphics kan ook streken en vullingen tekenen met verlopen in plaats van effen kleuren. Een verloopstreek wordt gemaakt met de methode `lineGradientStyle()` en een verloopvulling wordt gemaakt met de methode `beginGradientFill()`.

Beide methoden accepteren dezelfde parameters. De eerste vier zijn vereist: type, colors, alphas en ratios. De overige vier zijn optioneel, maar wel handig voor verdere aanpassing.

- De eerste parameter geeft het type verloop op dat u maakt. Acceptabele waarden zijn `GradientType.LINEAR` of `GradientType.RADIAL`.
- De tweede parameter geeft de array van de kleuren op die moeten worden gebruikt. In een lineair verloop worden de kleuren van links naar rechts geplaatst. In een radiaal verloop worden ze van binnen naar buiten geplaatst. De volgorde van de kleuren van de array staat voor de volgorde waarin de kleuren in het verloop worden getekend.
- De derde parameter geeft de alfatransparantiewaarden op van de bijbehorende kleuren in de vorige parameter.
- De vierde parameter geeft de verhouding of de nadruk op die elke kleur binnen het verloop krijgt. Acceptabele waarden lopen van 0 tot en met 255. Deze waarden geven geen breedte of hoogte aan, maar de positie binnen het verloop; 0 staat voor het begin van het verloop, 255 staat voor het einde van het verloop. De array met verhoudingen moet opeenvolgend oplopen en hetzelfde aantal items met informatie hebben als de kleur- en alfa-arrays die in de tweede en derde parameters zijn opgegeven.

Hoewel de vijfde parameter, de transformatiematrix, optioneel is, wordt deze vaak gebruikt omdat het een eenvoudige en krachtige manier is om het uiterlijk van het verloop te regelen. Deze parameter accepteert een instantie Matrix. De eenvoudigste manier om een object Matrix voor een verloop te maken is om de methode `createGradientBox()` van de klasse Matrix te gebruiken.

Een object Matrix definiëren voor gebruik met een verloop

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruik de methoden `beginGradientFill()` en `lineGradientStyle()` van de klasse flash.display.Graphics om verlopen te definiëren die in vormen moeten worden gebruikt. Wanneer u een verloop definieert, geeft u een matrix op als een van de parameters van deze methoden.

De eenvoudigste manier om de matrix te definiëren is om de methode `createGradientBox()` van de klasse Matrix te gebruiken; hierbij wordt een matrix gemaakt die wordt gebruikt om het verloop te definiëren. U definieert de schaal, rotatie en positie van het verloop met de parameters die aan de methode `createGradientBox()` zijn doorgegeven. De methode `createGradientBox()` accepteert de volgende parameters:

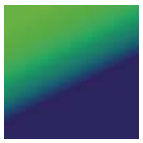
- Breedte verloopvak: de breedte (in pixels) waarover het verloop zich verspreidt.
- Hoogte verloopvak: de hoogte (in pixels) waarover het verloop zich verspreidt.
- Rotatie verloopvak: de rotatie (in radialen) die op het verloop moet worden toegepast.
- Horizontale translatie: hoe ver (in pixels) het verloop horizontaal verschuift.
- Verticale translatie: hoe ver (in pixels) het verloop verticaal verschuift.

Bekijk bijvoorbeeld een verloop met de volgende kenmerken:

- `GradientType.LINEAR`
- Twee kleuren, groen en blauw, met de array `ratios` ingesteld op `[0, 255]`
- `SpreadMethod.PAD`
- `InterpolationMethod.LINEAR_RGB`

In de volgende voorbeelden ziet u verlopen waarin de parameter `rotation` van de methode `createGradientBox()` afwijkt zoals aangegeven, maar alle andere instellingen hetzelfde blijven:

`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;`
	

In de volgende voorbeelden worden het effect getoond op een groen naar blauw, lineair verloop waarin de parameters `rotation`, `tx` en `ty` van de methode `createGradientBox()` afwijken zoals aangegeven, maar alle andere instellingen hetzelfde blijven:

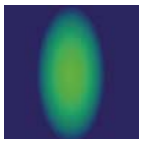
`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`
	

`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;`
	

De parameters `width`, `height`, `tx` en `ty` van de methode `createGradientBox()` hebben ook invloed op de grootte en positie van een verloopvulling *radial*, zoals u in het volgende voorbeeld ziet:

`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 25;`
`ty = 0;`
	

De volgende code produceert het laatste radiale verloop dat wordt weergegeven:

```
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);
```

De breedte en hoogte van de verloopvulling wordt bepaald door de breedte en hoogte van de verloopmatrix in plaats van door de breedte of hoogte die met het object Graphics is getekend. Wanneer u met het object Graphics tekent, tekent u wat er bij die coördinaten in de verloopmatrix bestaat. Zelfs als u een van de methoden shape van een object Graphics gebruikt, zoals `drawRect()`, rekt het verloop niet uit tot de grootte van de vorm die wordt getekend; de grootte van het verloop moet in de verloopmatrix zelf worden opgegeven.

In het volgende voorbeeld ziet u het visuele verschil tussen de afmetingen van de verloopmatrix en de afmetingen van de tekening zelf.

```
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);
```

Deze code tekent drie verlopen met dezelfde vullingstijl, die is opgegeven met een gelijke verdeling van rood, groen en blauw. De verlopen worden getekend met de methode `drawRect()` met pixelbreedten van respectievelijk 50, 100 en 150. De verloopmatrix die wordt opgegeven in de methode `beginGradientFill()` wordt gemaakt met een breedte van 100 pixels. Dit houdt in dat het eerste verloop slechts de helft van het verloopspectrum bevat, het tweede verloop alles bevat en het derde verloop alles bevat en daarbij 50 pixels blauw extra heeft die naar rechts uitwijken.

De methode `lineGradientStyle()` werkt net als `beginGradientFill()`, behalve dat niet alleen het verloop moet worden gedefinieerd, maar dat de dikte van de streek ook moet worden opgegeven met de methode `lineStyle()` voordat u begint met tekenen. De volgende code tekent een vak met een rode, groene en blauwe verloopstreek:

```
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);
```

Zie “Matrix-objecten gebruiken” op pagina 229 voor meer informatie over de klasse Matrix

De klasse Math gebruiken met tekenmethoden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een object Graphics tekent cirkels en vierkanten, maar kan ook meer ingewikkelde vormen tekenen, vooral wanneer de tekenmethoden worden gebruikt in combinatie met de eigenschappen en methoden van de klasse Math. De klasse Math bevat constanten die veel voorkomen in wiskundige formules, zoals `Math.PI` (ongeveer 3,14159265...), een constante voor de verhouding van de omtrek van een cirkel ten opzichte van de diameter. Het bevat ook methoden voor trigonometriefuncties, onder meer `Math.sin()`, `Math.cos()` en `Math.tan()`. Vormen die met deze methoden en constanten worden getekend, zorgen dat de visuele effecten dynamischer zijn, vooral wanneer ze met herhaling of recursie worden gebruikt.

Veel methoden van de klasse Match verwachten circulaire metingen in eenheden van radialen in plaats van graden. Omzetting tussen deze twee soorten eenheden wordt veel gedaan met de klasse Math:

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

In het volgende voorbeeld worden een sinusgolf en een cosinusgolf gemaakt om het verschil tussen de methoden `Math.sin()` en `Math.cos()` voor een bepaalde waarde aan te geven.

```
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);
}
```

Animaties met de teken-API

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een voordeel van het maken van inhoud met de teken-API is dat u zich niet hoeft te beperken tot het eenmalig plaatsen van de inhoud. Wat u tekent, kan worden aangepast door de variabelen die u gebruikt bij het tekenen te onderhouden en aan te passen. U kunt een animatie overbrengen door de variabelen te wijzigen en opnieuw te tekenen, gedurende een periode van frames of met een timer.

De volgende code wijzigt bijvoorbeeld de weergave bij elk doorgegeven frame (door naar de gebeurtenis `Event.ENTER_FRAME` te luisteren), waarbij de huidige gradentelling toeneemt, en laat het object graphics wissen en met de bijgewerkte positie opnieuw tekenen.

```
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;
}
```

U kunt een heel ander resultaat produceren door te experimenteren met het wijzigen van de beginwaarden voor de variabelen aan het begin van de code (`currentDegrees`, `radius` en `satelliteRadius`). U kunt bijvoorbeeld proberen de radiusvariabele te verkleinen en/of de variabele totalSatellites te verhogen. Dit is slechts één voorbeeld van hoe de teken-API een visuele weergave kan creëren waarbij de complexiteit de eenvoud van de creatie verbergt.

Voorbeeld van de API voor tekenen: Algorithmic Visual Generator

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het voorbeeld Algorithmic Visual Generator tekent dynamisch verschillende 'satellieten' of cirkels die in een cirkelvormige baan bewegen in het werkgebied. De volgende functies worden behandeld:

- De teken-API gebruiken om een basisvorm te tekenen met een dynamisch uiterlijk.
- Gebruikersinteractie koppelen aan de eigenschappen die in een tekening worden gebruikt.
- Animatie overbrengen door het werkgebied op elk frame te wissen en opnieuw te tekenen.

In het voorbeeld in de vorige subsectie werd één 'satelliet' bewegend gemaakt met de gebeurtenis `Event.ENTER_FRAME`. Dit wordt hier uitgebreid door een bedieningspaneel te maken met een reeks schuifregelaars die de visuele weergave van verschillende satellieten direct bijwerken. In dit voorbeeld wordt de code in externe klassen geformaliseerd en wordt de code voor het maken van de satelliet in een lus geplaatst, waarbij een verwijzing naar elke satelliet in een array `satellites` wordt opgeslagen.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden vindt u in de map Samples/AlgorithmicVisualGenerator. In deze map staan de volgende bestanden:

Bestand	Beschrijving
AlgorithmicVisualGenerator.fla	Het hoofdtoepassingsbestand in Flash Professional (FLA).
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	De klasse met de hoofdfunctionaliteit van de toepassing, inclusief het tekenen van satellieten in het werkgebied en het reageren op gebeurtenissen van het bedieningspaneel zodat de variabelen worden bijgewerkt die effect hebben op het tekenen van satellieten.
com/example/programmingas3/algorithmic/ControlPanel.as	Een klasse die de gebruikersinteractie beheert met verschillende schuifregelaars en gebeurtenissen verzendt wanneer dergelijke interactie zich voordoet.
com/example/programmingas3/algorithmic/Satellite.as	Een klasse die staat voor het weergaveobject dat in een baan rond een centraal punt draait en dat eigenschappen bevat die betrekking hebben op de huidige tekenstatus.

Listeners instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing maakt eerst drie listeners. De eerste luistert naar een door het bedieningspaneel verzonden gebeurtenis die aangeeft dat de satellieten opnieuw moeten worden opgebouwd. De tweede luistert naar wijzigingen in de grootte van het werkgebied voor het SWF-bestand. De derde luistert naar elk doorgegeven frame in het SWF-bestand en gebruikt de functie `doEveryFrame()` om opnieuw te tekenen.

Satellieten maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer deze luisteraars zijn ingesteld, wordt de functie `build()` aangeroepen. Deze functie roept eerst de functie `clear()` aan, die de array `satellites` leegt en eventuele eerdere tekenhandelingen in het werkgebied wist. Dit is nodig aangezien de functie `build()` opnieuw kan worden aangeroepen wanneer het bedieningspaneel hiervoor een gebeurtenis verzendt, bijvoorbeeld wanneer de kleurinstellingen zijn gewijzigd. In dat geval moeten de satellieten worden verwijderd en opnieuw gemaakt.

De functie maakt daarna de satellieten en stelt daarbij de eerste eigenschappen in die nodig zijn voor het maken ervan, zoals de variabele `position`, die op een willekeurige positie in de baan begint en de variabele `color`, die in dit voorbeeld niet meer wordt gewijzigd wanneer de satelliet is gemaakt.

Na het maken van elke satelliet, wordt een verwijzing naar de satelliet toegevoegd aan de array `satellites`. Wanneer de functie `doEveryFrame()` wordt aangeroepen, worden alle satellieten in deze array bijgewerkt.

Positie van de satellieten bijwerken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De functie `doEveryFrame()` is de kern van het animatieproces van de toepassing. Deze wordt voor elk frame aangeroepen, met een snelheid die gelijk staat aan de framesnelheid van het SWF-bestand. Omdat de variabelen voor het tekenen iets wijzigen, geeft dit het idee van een animatie.

De functie wist eerst alle eerdere tekenhandelingen en tekent de achtergrond opnieuw. Vervolgens loopt deze elke satellietcontainer door en wordt de eigenschap `position` van elke satelliet verhoogd. De eigenschappen `radius` en `orbitRadius` worden dan bijgewerkt omdat die mogelijk zijn gewijzigd door gebruikersinteractie met het bedieningspaneel. Ten slotte wordt de satelliet op de nieuwe positie ingesteld door de methode `draw()` van de klasse Satellite aan te roepen.

De teller `i` neemt alleen toe tot de variabele `visibleSatellites`. De reden hiervoor is dat wanneer de gebruiker het aantal satellieten heeft beperkt die via het bedieningspaneel worden weergegeven, de overige satellieten in de lus niet opnieuw hoeven te worden getekend, maar verborgen moeten worden. Dit gebeurt in een lus die onmiddellijk volgt op de lus die verantwoordelijk is voor het tekenen.

Wanneer de functie doEveryFrame() is voltooid, wordt het aantal `visibleSatellites` op hun nieuwe positie op het scherm ingesteld.

Reageren op gebruikersinteractie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruikersinteractie doet zich voor via het bedieningspaneel, dat door de klasse ControlPanel wordt beheerd. Deze klasse stelt een listener in samen met de afzonderlijke minimale, maximale en standaardwaarden voor elke schuifregelaar. Wanneer de gebruiker deze schuifregelaars verplaatst, wordt de functie `changeSetting()` aangeroepen. Deze functie werkt de eigenschappen van het bedieningspaneel bij. Wanneer de weergave bij het wijzigen opnieuw moet worden opgebouwd, wordt een gebeurtenis verzonden die daarna in het hoofdtoepassingsbestand wordt verwerkt. Wanneer de instellingen van het bedieningspaneel worden gewijzigd, tekent de functie `doEveryFrame()` elke satelliet met de bijgewerkte variabelen.

Verdere aanpassingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Dit voorbeeld is slechts een eenvoudig overzicht van hoe u visuele effecten kunt maken met behulp van de teken-API. Er worden relatief weinig coderegels gebruikt om een interactieve ervaring te maken die er ingewikkeld uitziet. Toch kan dit voorbeeld nog verder worden uitgebreid. Enkele ideeën:

- De functie `doEveryFrame()` kan de kleurwaarde van de satelliet verhogen.
- De functie `doEveryFrame()` kan de satellietradius gedurende een periode verkleinen of vergroten.
- De satellietradius hoeft niet circulair te zijn; de klasse Math kan bijvoorbeeld worden gebruikt om volgens een sinusgolf te bewegen.
- Satellieten kunnen aanraakdetectie met andere satellieten gebruiken.

De teken-API kan worden gebruikt als een alternatief voor het maken van visuele effecten in de Flash-ontwerpomgeving door basisvormen bij uitvoering te tekenen. Maar de teken-API kan ook worden gebruikt om visuele effecten te maken met een variatie en bereik die niet met de hand kunnen worden bereikt. Met de teken-API en een beetje wiskunde kan de ActionScript-ontwerper veel onverwachte creaties maken.

Geavanceerd gebruik van de teken-API

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Flash Player 10, Adobe AIR 1.5 en latere Flash-runtimes ondersteunen een geavanceerde set tekenfuncties. De API-uitbreidingen voor tekenen voor deze runtimes bouwen verder op de tekenmethoden van vorige releases zodat u gegevenssets kunt maken om vormen te genereren, vormen tijdens runtime te wijzigen en 3D-effecten te maken. De API-uitbreidingen voor tekenen verenigen bestaande methoden in alternatieve opdrachten. Deze opdrachten bieden vectorarrays en opsommingsklassen om gegevenssets voor tekenmethoden te leveren. Met vectorarrays zijn complexere vormen mogelijk die sneller gerenderd kunnen worden. Ontwikkelaars kunnen de arraywaarden via programmacode wijzigen voor dynamische rendering van vormen tijdens de runtime.

De tekenfuncties die in Flash Player 10 zijn geïntroduceerd, worden in de volgende secties beschreven: “Tekenpaden” op pagina 249, “Wentelregels definiëren” op pagina 250, “Grafische gegevensklassen gebruiken” op pagina 252 en “Informatie over drawTriangles()” op pagina 255.

De volgende taken wilt u wellicht uitvoeren met behulp van de geavanceerde API voor tekenen in ActionScript:

- Vector-objecten gebruiken om gegevens voor tekenmethoden op te slaan
- Paden definiëren voor het met programmacode tekenen van vormen in één bewerking
- Wentelregels definiëren om te bepalen hoe overlappende vormen worden gevuld
- De inhoud van de vectorafbeeldingen van een weergaveobject lezen, zoals het serialiseren en opslaan van de grafische gegevens, om een sprite-werkblad bij uitvoering te genereren of om een kopie van de inhoud van de vectorafbeeldingen te tekenen
- Driehoeken en tekenmethoden gebruiken voor driedimensionale effecten

Belangrijke concepten en termen

De volgende lijst bevat belangrijke termen die u in deze sectie zult tegenkomen:

- Vector: een array met waarden van hetzelfde gegevenstype. Een Vector-object kan een array opslaan met waarden die door tekenmethoden worden gebruikt om met één opdracht lijnen en vormen te tekenen. Zie "Geïndexeerde arrays" op pagina 28 voor meer informatie over Vector-objecten.
- Pad: een pad bestaat uit een of meer rechte of gekromde segmenten. Het begin en einde van elk segment worden gemarkeerd door coördinaten, die werken als spelden die een draad op zijn plaats houden. Een pad kan gesloten (bijvoorbeeld een cirkel) zijn of open, met duidelijke eindpunten (bijvoorbeeld een golvende lijn).
- Wenteling: de richting van een pad, zoals deze wordt geïnterpreteerd door de renderer; dus positief (rechtsom) of negatief (linksom).
- GraphicsStroke: een klasse waarmee de lijnstijl wordt ingesteld. De term 'streek' maakt geen deel uit van de API-uitbreidingen voor tekenen, maar het gebruik van een klasse om een stijl van een lijn met een eigen vuleigenschap in te stellen maakt wel deel uit van de nieuwe API voor tekenen. U kunt de stijl van een lijn dynamisch wijzigen met de klasse GraphicsStroke.
- Fill-object: objecten die met nieuwe weergaveklassen zijn gemaakt, bijvoorbeeld met flash.display.GraphicsBitmapFill en flash.display.GraphicsGradientFill, en worden doorgegeven aan de tekenopdracht `Graphics.drawGraphicsData()`. Fill-objecten en de uitgebreide tekenopdrachten introduceren een meer objectgeoriënteerde programmeerbenadering ten opzichte van het repliceren van `Graphics.beginBitmapFill()` en `Graphics.beginGradientFill()`.

Tekenpaden

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In de sectie over het tekenen van lijnen en curven (zie "Lijnen en curven tekenen" op pagina 236) zijn de opdrachten geïntroduceerd voor het tekenen van één lijn (`Graphics.lineTo()`) of curve (`Graphics.curveTo()`) en het verplaatsen van de lijn naar een ander punt (`Graphics.moveTo()`) om een vorm te creëren. De methoden `Graphics.drawPath()` en `Graphics.drawTriangles()` accepteren een set met objecten die dezelfde tekenopdrachten vertegenwoordigen als een parameter. Met deze methoden kunt u een reeks opdrachten `Graphics.lineTo()`, `Graphics.curveTo()` of `Graphics.moveTo()` opgeven die de Flash-runtime in één instructie kan uitvoeren.

De opsommingsklasse GraphicsPathCommand definieert een set met constanten die overeenstemmen met tekenopdrachten. U stuurt een reeks van deze constanten (ingepakt in een Vector-instantie) door als een parameter voor de `Graphics.drawPath()`-methode. Daarna kunt u met één opdracht een gehele vorm of meerdere vormen renderen. U kunt ook de waarden wijzigen die aan deze methoden zijn doorgegeven, om een bestaande vorm te wijzigen.

Naast de Vector met tekenopdrachten heeft de `drawPath()`-methode een set met coördinaten nodig die overeenstemmen met de coördinaten voor elke tekenopdracht. Maak een Vector-instantie die coördinaten (Number-instanties) bevat en stuur deze door naar de `drawPath()`-methode als het tweede (`data`)-argument.

Opmerking: *De waarden in de vector zijn geen Point-objecten. De vector is een reeks getallen waarbij telkens twee getallen een x/y-coördinaat vormen.*

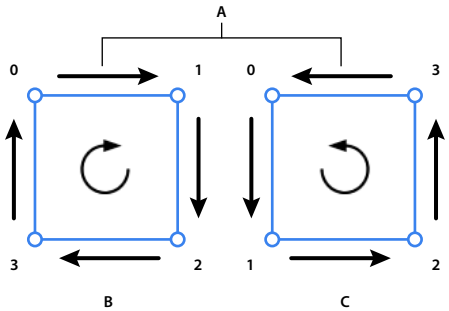
De methode `Graphics.drawPath()` vergelijkt elke opdracht met de respectieve puntwaarden (een verzameling van twee of vier getallen) om een pad te genereren in het Graphics-object:

```
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);
        }
    }
}
```

Wentelregels definiëren

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De API-uitbreiding voor tekenen introduceert ook het concept "wenteling" van een pad: de richting van een pad. De wenteling van een pad is positief (rechtsom) of negatief (linksom). De wenteling wordt bepaald door de volgorde waarin de renderer de coördinaten interpreteert die door de vector voor de gegevensparameter worden verstrekt.



Positieve en negatieve wenteling
***A.** Pijlen geven de tekenrichting aan **B.** Positief gewenteld (rechtsom) **C.** Negatief gewenteld (linksom)*

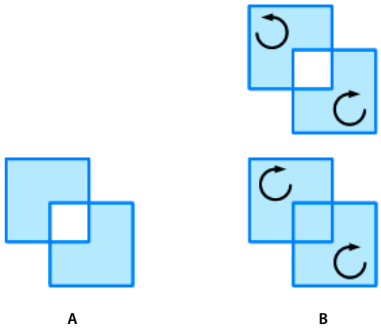
U ziet bovendien dat de methode `Graphics.drawPath()` een optionele derde parameter heeft, de parameter 'winding':

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

In deze context is de derde parameter een tekenreeks of constante die de wenteling of vulregel bepaalt voor paden die elkaar kruisen. (De constante waarden worden in de klasse GraphicsPathWinding gedefinieerd als `GraphicsPathWinding.EVEN_ODD` of `GraphicsPathWinding.NON_ZERO`.) De wentelregel is belangrijk op het moment dat paden elkaar kruisen.

De even/oneven-regel is de standaardwentelregel. Deze regel wordt gebruikt door de verouderde API voor tekenen. De even/oneven-regel is ook de standaardregel voor de methode `Graphics.drawPath()`. Met de even/oneven-wentelregel wisselen elkaar kruisende paden tussen open en gesloten vullingen. Als twee getekende vierkanten met dezelfde vulling elkaar snijden, wordt het gebied van de doorsnede niet gevuld. Over het algemeen worden naast elkaar gelegen gebieden niet allebei gevuld of ongevuld.

De niet-nulregel is daarentegen afhankelijk van de wenteling (tekenrichting) bij de bepaling of gebieden die door elkaar kruisende paden worden gedefinieerd, al dan niet worden gevuld. Als paden van tegenovergestelde wentelingen elkaar kruisen, wordt het gedefinieerde gebied niet gevuld, ongeveer zoals bij even/oneven. Voor paden met dezelfde wenteling wordt het gebied dat niet gevuld zou zijn, gevuld:

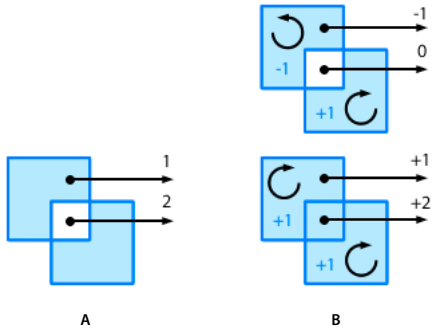


Wentelregels voor elkaar kruisende gebieden
***A.** Even/oneven-regel **B.** Niet-nulregel*

Namen van wentelregels

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De namen verwijzen naar een specifiekere regel die definieert hoe deze vullingen worden beheerd. Positief gewentelde paden krijgen de waarde +1; negatief gewentelde paden krijgen de waarde -1. Teken een lijn die zich oneindig uitstrekt vanaf een punt binnen een omsloten gebied van een vorm. De vulling wordt bepaald aan de hand van het aantal keer dat de lijn een pad kruist, en de gecombineerde waarden van deze paden. Voor even/oneven-wenteling wordt het aantal keer dat de lijn een pad kruist, gebruikt. Als het aantal keer oneven is, wordt het gebied gevuld. Voor even aantallen wordt het gebied niet gevuld. Voor niet-nulwentelingen worden de waarden gebruikt die aan de paden zijn toegewezen. Als de gecombineerde waarde van het pad niet 0 is, wordt het gebied gevuld. Als de gecombineerde waarde 0 is, wordt het gebied niet gevuld.



Aantallen en vullingen voor wentelregels
***A.** Even/oneven-regel **B.** Niet-nulregel*

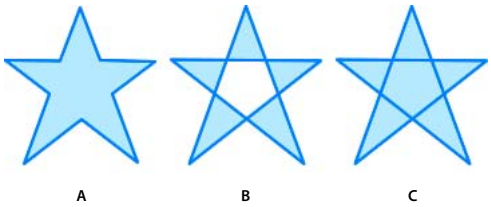
Wentelregels gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Deze vulregels zijn ingewikkeld, maar in sommige gevallen zijn ze noodzakelijk. Neem bijvoorbeeld een stervorm. Met de standaard even/oneven-regel zou de vorm tien verschillende lijnen nodig hebben. Met de niet-nulwentelregel worden deze tien lijnen gereduceerd tot vijf. Hier volgt de ActionScript-code voor een ster met vijf lijnen en een niet-nulwentelregel:

```
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);
```

En hier is de stervorm:



Een stervorm met verschillende wentelregels
***A.** Even/oneven 10 lijnen **B.** Even/oneven 5 lijnen **C.** Niet-nul 5 lijnen*

Wanneer afbeeldingen worden geanimeerd of als structuur voor driedimensionale objecten worden gebruikt en elkaar overlappen, worden de wentelregels belangrijker.

Grafische gegevensklassen gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De API-uitbreiding voor tekenen bevat een set met klassen in het flash.display-pakket die de IGraphicsData-interface implementeren. Deze klassen fungeren als waardeobjecten (gegevenscontainers) die de tekenmethoden van de teken-API vertegenwoordigen.

De volgende klassen implementeren de IGraphicsData-interface:

- GraphicsBitmapFill
- GraphicsEndFill

- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

Met deze klassen kunt u een volledige tekening opslaan in een Vector-object van het type IGraphicsData (Vector.<IGraphicsData>). U kunt de grafische gegevens opnieuw gebruiken als de gegevensbron voor andere vorminstanties of om tekeninformatie op te slaan voor later gebruik.

U ziet dat er meerdere vulklassen voor elke vulstijl zijn, maar slechts één streekklasse. ActionScript heeft maar één IGraphicsData-streekklasse omdat de streekklasse de vulklassen gebruikt om de stijl ervan te definiëren. Elke streek wordt in feite dus gedefinieerd door een combinatie van de streekklasse en een vulklasse. De API's voor deze afbeeldingsgegevensklassen spiegelen overigens de methoden die ze in de klasse flash.display.Graphics vertegenwoordigen:

Graphics-methode	**Overeenstemmende klasse**
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

Daarnaast heeft de klasse GraphicsPath zijn eigen hulpprogrammamethoden `GraphicsPath.moveTo()`, `GraphicsPath.lineTo()`, `GraphicsPath.curveTo()`, `GraphicsPath.wideLineTo()` en `GraphicsPath.wideMoveTo()`, zodat deze opdrachten op eenvoudige wijze voor een GraphicsPath-instantie kunnen worden gedefinieerd. Deze hulpprogrammamethoden vereenvoudigen het rechtstreeks definiëren en bijwerken van de opdrachten en gegevenswaarden.

Tekenen met gegevens van vectorafbeeldingen

Wanneer u een verzameling IGraphicsData-instanties hebt, gebruikt u de `drawGraphicsData()`-methode van de Graphics-klasse om de afbeeldingen te renderen. De `drawGraphicsData()`-methode voert een set met tekeninstructies in sequentiële volgorde uit vanaf een vector van IGraphicsData-instanties:

```
// 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);
```

Door één waarde te wijzigen in het pad dat door de tekening in het voorbeeld wordt gebruikt, kan de vorm meerdere keren opnieuw worden getekend om een complexere afbeelding te verkrijgen:

```
// 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);
```

Ondanks het feit dat IGraphicsData-objecten vul- en streekstijlen kunnen definiëren, zijn de vul- en streekstijlen geen vereiste. Met andere woorden, methoden van de klasse Graphics kunnen worden gebruikt om stijlen in te stellen, terwijl IGraphicsData-objecten kunnen worden gebruikt om een opgeslagen verzameling paden te tekenen, of omgekeerd.

Opmerking: *Gebruik de methode* `Graphics.clear()` *om een vorige tekening te wissen voordat u een nieuwe maakt, tenzij u de oorspronkelijke tekening wilt uitbreiden, zoals in het voorbeeld hierboven. Terwijl u één gedeelte van een pad of verzameling van IGraphicsData-objecten wijzigt, kunt u de hele tekening opnieuw tekenen om de wijzigingen te zien.*

Wanneer u grafische gegevensklassen gebruikt, wordt de vulling gerenderd wanneer drie of meer punten worden getekend, aangezien de vorm zich inherent op dat punt sluit. De vulling wordt gesloten, maar de streek niet. Dit gedrag is anders wanneer meerdere `Graphics.lineTo()`- of `Graphics.moveTo()`-opdrachten worden gebruikt.

Gegevens van vectorafbeeldingen lezen

Flash Player 11.6 of hoger, Adobe AIR 3.6 of hoger

Naast het tekenen van vectorinhoud naar een weergaveobject kunt u in Flash Player 11.6 en Adobe AIR 3.6 en later de `readGraphicsData()`-methode van de Graphics-klasse gebruiken om een gegevensrepresentatie te verkrijgen van de inhoud van de vectorafbeeldingen van een weergaveobject. Deze kan worden gebruiken om een momentopname van een afbeelding te maken om een sprite-werkblad bij uitvoering op te slaan, te kopiëren, te maken en meer.

Door de `readGraphicsData()`-methode aan te roepen, wordt een Vector-instantie geretourneerd die IGraphicsData-objecten bevat. Dit zijn dezelfde objecten die worden gebruikt om vectorafbeeldingen te tekenen met de `drawGraphicsData()`-methode.

Er zijn verschillende beperkingen voor het lezen van vectorafbeeldingen met de `readGraphicsData()`-methode. Zie de readGraphicsData()-vermelding in de Naslaggids voor ActionScript.

Informatie over drawTriangles()

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Een andere geavanceerde methode die in Flash Player 10 en Adobe AIR 1.5 is geïntroduceerd, namelijk `Graphics.drawTriangles()`, lijkt op de methode `Graphics.drawPath()`. De methode `Graphics.drawTriangles()` gebruikt eveneens een Vector.<Number>-object om puntlocaties voor het tekenen van paden op te geven.

De methode `Graphics.drawTriangles()` is echter werkelijk bedoeld om driedimensionale effecten te vereenvoudigen middels ActionScript. Voor informatie over het gebruik van `Graphics.drawTriangles()` om driedimensionale effecten te produceren, raadpleegt u “Driehoeken gebruiken voor 3D-effecten” op pagina 383.

Hoofdstuk 13: Werken met bitmaps

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naast vectortekeningen kunt u in ActionScript 3.0 bitmapafbeeldingen maken en de pixelgegevens bewerken van externe bitmapafbeeldingen die zijn geladen in een SWF-bestand. Omdat u afzonderlijke pixelwaarden kunt wijzigen, kunt u uw eigen filterachtige afbeeldingseffecten maken en de ingebouwde ruisfuncties gebruiken om structuren en willekeurige ruis te maken.

- Renaun Erickson: Game-assets renderen in ActionScript met behulp van 'blitting'-methoden
- Bitmap programming: Hoofdstuk 26 van Essential ActionScript 3 door Colin Moock (O'Reilly Media, 2007)
- Mike Jones: Working with Sprites in Pushbutton Engine
- Flash & Math: Pixel Particles Made Simple
- Flixel

Basisbeginselen van het werken met bitmaps

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u met digitale afbeeldingen werkt, komt u doorgaans waarschijnlijk twee typen afbeeldingen tegen: bitmap en vector. Bitmapafbeeldingen, ook wel rasterafbeeldingen genoemd, bestaan uit kleine vierkantjes (pixels) die in een rechthoekige rasterformatie zijn gerangschikt. Vectorafbeeldingen bestaan uit wiskundig gegenereerde geometrische vormen als lijnen, curven en veelhoeken.

Bitmapafbeeldingen worden gedefinieerd op basis van de breedte en hoogte van de afbeelding, gemeten in pixels, en het aantal bits in elke pixel, waarmee wordt aangegeven hoeveel kleuren een pixel kan bevatten. In een bitmapafbeelding waarin wordt gebruikgemaakt van het RGB-kleurmodel bestaan de pixels uit drie bytes: rood, groen en blauw. Elk van deze bytes bevat een waarde variërend van 0 tot 255. Wanneer de bytes in de pixel worden gecombineerd, wordt een kleur geproduceerd op een wijze vergelijkbaar met het mengen van verfkleuren. Een pixel met de bytewaarden rood-255, groen-102 en blauw-0 resulteert bijvoorbeeld in een levendige oranje kleur.

De kwaliteit van een bitmapafbeelding is afhankelijk van de resolutie van de afbeelding en de bitwaarde voor de kleurdiepte. *Resolutie* heeft betrekking op het aantal pixels in een afbeelding. Hoe groter het aantal pixels, hoe hoger de resolutie en hoe scherper de afbeelding. *Kleurdiepte* heeft betrekking op de hoeveelheid gegevens die een pixel kan bevatten. In een afbeelding met een kleurdieptewaarde van 16 bits per pixel kunnen bijvoorbeeld niet evenveel kleuren worden gebruikt als in een afbeelding met een kleurdiepte van 48 bits. Als gevolg daarvan is het verloop in de 48-bits afbeelding vloeiender dan in de 16-bits afbeelding.

Omdat bitmapafbeeldingen resolutieafhankelijk zijn, kunnen ze niet goed worden geschaald. Dit valt vooral op wanneer bitmapafbeelding worden vergroot. Wanneer een bitmap wordt vergroot, gaat dit doorgaans ten koste van de gedetailleerdheid en kwaliteit.

Bestandsindelingen voor bitmaps

Bitmapafbeeldingen worden gegroepeerd in een aantal gangbare bestandsindelingen. In deze indelingen worden verschillende typen compressiealgoritmes gebruikt om de bestandsgrootte te beperken en de afbeeldingskwaliteit te optimaliseren op basis van het uiteindelijke doel van de afbeelding. De bitmapafbeeldingsindelingen die worden ondersteund door Adobe-runtimes zijn BMP, GIF, JPG, PNG, and TIFF.

BMP

De indeling BMP (bitmap) is de standaardindeling voor afbeeldingen in het besturingssysteem Microsoft Windows. Hierin worden geen compressiealgoritmen gebruikt; dit leidt daarom gewoonlijk tot grotere bestandsomvangen.

GIF

De GIF-indeling (Graphics Interchange Format) is in 1987 door CompuServe ontwikkeld om afbeeldingen met 256 kleuren (8-bits kleur) te kunnen verzenden. De bestandsgrootte van deze afbeeldingen is klein en is ideaal voor webafbeeldingen. Vanwege het beperkte kleurenpalet van deze indeling zijn GIF-afbeeldingen doorgaans niet geschikt voor foto's, waarvoor over het algemeen een hoge mate van schaduw- en kleurverloop vereist is. In GIF-afbeeldingen is 1-bits transparantie mogelijk, zodat kleuren doorzichtig (of transparant) kunnen worden weergegeven. In dat geval is de achtergrondkleur van de webpagina in de gebieden die zijn ingesteld als transparant door de afbeelding heen te zien.

JPEG

De bestandsindeling JPEG (vaak geschreven als JPG) is ontwikkeld door en genoemd naar de Joint Photographic Experts Group en gebruikt een algoritme voor compressie met verlies om 24-bits kleurdiepte in bestanden met een beperkte bestandsgrootte mogelijk te maken. Compressie met verlies betekent dat wanneer het bestand wordt opgeslagen, de kwaliteit van de afbeelding minder wordt en er gegevens verloren gaan. De JPEG-indeling is ideaal voor foto's omdat hierin miljoenen kleuren kunnen worden weergegeven. Omdat u kunt bepalen welk compressieniveau wordt toegepast op de afbeelding, hebt u invloed op de afbeeldingskwaliteit en bestandsgrootte.

PNG

DE PNG-indeling (Portable Network Graphics) is ontwikkeld als open-bronalternatief voor de gepatenteerde bestandsindeling GIF. PNG-bestanden bieden ondersteuning voor een kleurdiepte van maximaal 64 bits en 16 miljoen kleuren. Omdat PNG een relatief nieuwe indeling is, worden PNG-bestanden in sommige oudere browsers niet ondersteund. In tegenstelling tot JPG-bestanden wordt in PNG-bestanden gebruikgemaakt van compressie zonder verlies. Dit betekent dat er geen gegevens verloren gaan wanneer de afbeelding wordt opgeslagen. PNG-bestanden bieden tevens ondersteuning voor alfatransparantie, zodat er 256 transparantieniveaus kunnen worden toegepast.

TIFF

De indeling TIFF (Tagged Image File Format) was de platformoverschrijdende keuze voordat PNG werd geïntroduceerd. Het nadeel van de indeling TIFF is dat geen enkele lezer elke versie kan verwerken omdat er zoveel varianten van TIFF bestaan. Bovendien wordt de indeling tot op heden door geen enkele webbrowser ondersteund. TIFF kan zowel compressie met als zonder verlies gebruiken, en kan met apparaatspecifieke kleurruimten (zoals CMYK) werken.

Transparante en dekkende bitmaps

In bitmapafbeeldingen in de GIF- of PNG-indeling kan nog een extra byte (alfakanaal) worden toegevoegd aan elke pixel. Deze extra pixelbyte wordt gebruikt voor de transparantiewaarde van de pixel.

GIF-afbeeldingen bieden ondersteuning voor 1-bits transparantie. Dit betekent dat u één transparante kleur kunt opgeven in een palet met 256 kleuren. In PNG-bestanden kunnen daarentegen 256 transparantieniveaus worden toegepast. Deze functie is met name nuttig wanneer een afbeelding of tekst moet overvloeien in de achtergrond.

ActionScript 3.0 dupliceert deze extra transparantiepixelbyte in de klasse BitmapData. Net als het PNG-transparantiemodel, biedt ActionScript tot 256 transparantieniveaus.

Belangrijke concepten en termen
De volgende lijst bevat belangrijke termen die u tegenkomt wanneer u leert over bitmapafbeeldingen:

Alfa Het transparantieniveau (of preciezer, opaciteit) in een kleur of afbeelding. De hoeveelheid alfa wordt vaak beschreven als de *alfakanaal*waarde.

ARGB-kleur Een kleurschema waarin de kleur van elke pixel een mengeling van de kleurwaarden rood, groen en blauw en de transparantie ervan wordt opgegeven als een alfawaarde.

Kleurkanaal Kleuren worden over het algemeen weergegeven als een mengeling van een aantal basiskleuren; gewoonlijk (voor computerafbeeldingen) rood, groen en blauw. Elke basiskleur wordt beschouwd als een kleurkanaal en de uiteindelijke kleur wordt bepaald op basis van de hoeveelheid kleur in de verschillende kanalen.

Kleurdiepte Dit staat ook bekend als *bitdiepte* en verwijst naar de hoeveelheid computergeheugen die aan elke pixel wordt toegewezen. Dit bepaalt het aantal mogelijke kleuren dat in de afbeeldingen kan worden weergegeven.

Pixel De kleinste gegevenseenheid in een bitmapafbeelding, in feite een gekleurde stip.

Resolutie De pixeldimensies van een afbeelding, die het niveau van fijnkorrelige details in de afbeelding bepaalt. De resolutie wordt vaak uitgedrukt in het aantal pixels in de breedte en hoogte.

RGB-kleur Een kleurschema waarin de kleur van elke pixel wordt weergegeven als een mengeling van rood-, groen- en blauwwaarden.

De klassen Bitmap en BitmapData

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De belangrijkste ActionScript 3.0-klassen voor het werken met bitmapafbeeldingen zijn de Bitmap-klasse, die wordt gebruikt voor de weergave van bitmapafbeeldingen op het scherm en de BitmapData-klasse, die wordt gebruikt voor het openen en manipuleren van de ruwe afbeeldingsgegevens van een bitmap.

Meer Help-onderwerpen
flash.display.Bitmap

flash.display.BitmapData

De klasse Bitmap

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als een subklasse van de klasse DisplayObject is de klasse Bitmap de belangrijkste ActionScript 3.0-klasse voor het weergeven van bitmapafbeeldingen. Deze afbeeldingen kunnen zijn geladen via de klasse flash.display.Loader of worden dynamisch gemaakt met de `Bitmap()`-constructor. Bij het laden van een afbeelding uit een externe bron kan een object Bitmap alleen afbeeldingen in de GIF-, JPEG- of PNG-indeling gebruiken. Wanneer deze is geïnstantieerd, kan de Bitmap-instantie worden beschouwd als een wrapper voor een object BitmapData dat moet worden gerenderd naar het werkgebied. Omdat een Bitmap-instantie een weergaveobject is, kunnen alle kenmerken en functies van weergaveobjecten ook worden gebruikt om een Bitmap-instantie te bewerken. Zie "Weergave programmeren" op pagina 161 voor meer informatie over het werken met weergaveobjecten.

Pixels uitlijnen en vloeiend maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naast de functies die alle weergaveobjecten gemeen hebben, biedt de klasse Bitmap specifieke functies voor bitmapafbeeldingen.

De eigenschap `pixelSnapping` van de klasse Bitmap bepaalt of een Bitmap-object op de dichstbijzijnde pixel is uitgelijnd. Deze eigenschap accepteert een van deze drie constanten die in de klasse PixelSnapping zijn gedefinieerd: `ALWAYS`, `AUTO` en `NEVER`.

De syntaxis voor het toepassen van pixeluitlijning is als volgt:

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

In veel gevallen worden bitmapafbeeldingen vaag en vervormd weergegeven wanneer ze worden geschaald. Met de eigenschap `smoothing` van de klasse BitmapData kunt u deze vervorming beperken. Wanneer deze Booleaanse eigenschap is ingesteld op `true`, worden de pixels in de afbeelding vloeiend gemaakt (anti-aliasing) wanneer deze wordt geschaald. Hierdoor wordt de afbeelding helderder en meer natuurgetrouw.

De klasse BitmapData

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse BitmapData, die zich in het pakket flash.display bevindt, kan worden vergeleken met een fotografische momentopname van de pixels in een geladen of dynamisch gemaakte bitmapafbeelding. Deze momentopname wordt weergegeven als een array van pixelgegevens in het object. De klasse BitmapData bevat ook een reeks ingebouwde methoden die nuttig zijn voor het maken en bewerken van pixelgegevens.

Gebruik de volgende code als u een object BitmapData wilt instantiëren:

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

Door de parameters `width` en `height` worden de afmetingen van de bitmap aangegeven. Vanaf AIR 3 en Flash Player 11 gelden er geen beperkingen meer voor de grootte van BitmapData-objecten. De maximale grootte van een bitmap is afhankelijk van het besturingssysteem.

In AIR 1.5 en Flash Player 10 is de maximale grootte voor een object BitmapData 8,191 pixels breed of hoog en mag het totale aantal pixels niet groter zijn dan 16,777,215 pixels. (Als een object BitmapData 8,191 pixels breed is, kan het daarom slechts 2048 pixels hoog zijn.) In Flash Player 9 en lager en AIR 1.1 en lager is de maximale hoogte 2880 en de maximale breedte 2880 pixels.

Met de parameter `transparent` geeft u op of in bitmapgegevens al dan niet een alfakanaal moet worden opgenomen (`true` of `false`). De parameter `fillColor` is een 32-bits kleurwaarde waarmee de achtergrondkleur en de transparantiewaarde (als deze is ingesteld op `true`) worden opgegeven. In het volgende voorbeeld wordt een object BitmapData met een oranje achtergrond gemaakt die voor 50 procent transparant is.

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

Als u een nieuw object BitmapData op het scherm wilt renderen, moet u het toewijzen aan of opnemen in een Bitmap-instantie. U kunt dit doen door het object BitmapData als een parameter van de constructor van het object Bitmap door te geven of toe te wijzen aan de eigenschap `bitmapData` van een bestaande Bitmap-instantie. Daarnaast moet u de Bitmap-instantie aan het weergaveoverzicht toevoegen door de methode `addChild()` of `addChildAt()` aan te roepen van de weergaveobjectcontainer die de Bitmap-instantie bevat. Zie "Weergaveobjecten aan het weergaveoverzicht toevoegen" op pagina 169 voor meer informatie over het werken met het weergaveoverzicht.

In het volgende voorbeeld wordt een object BitmapData met een rode vulling gemaakt dat wordt weergegeven in een Bitmap-instantie.

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

Pixels bewerken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse BitmapData bevat een set methoden waarmee u pixelgegevenswaarden kunt bewerken.

Afzonderlijke pixels bewerken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de weergave van een bitmapafbeelding op pixelniveau wilt wijzigen, moet u eerst de kleurwaarden ophalen van de pixels in het gebied dat u wilt bewerken. U gebruikt de methode `getPixel()` om deze pixelwaarden te lezen.

Met de methode `getPixel()` wordt een RGB-waarde opgehaald uit een set x- en y-(pixel)coördinaten die worden doorgegeven als een parameter. Als een van de pixels die u wilt bewerken transparantiegegevens (alfakanaal) bevat, moet u de methode `getPixel32()` gebruiken. Met deze methode wordt ook een RGB-waarde opgehaald, maar in tegenstelling tot met `getPixel()`, bevat de door `getPixel32()` geretourneerde waarde aanvullende gegevens met betrekking tot de alfakanaalwaarde (transparantie) van de geselecteerde pixel.

Als u slechts de kleur of transparantie van een pixel in een bitmap wilt wijzigen, kunt u de methode `setPixel()` of `setPixel32()` gebruiken. U stelt de kleur van een pixel in door de x- en y-coördinaten en de kleurwaarde door te geven aan een van deze methoden.

In het volgende voorbeeld wordt `setPixel()` gebruikt om een kruis op een groene BitmapData-achtergrond te tekenen. Vervolgens wordt `getPixel()` gebruikt om de kleurwaarde van de pixel op coördinaat 50, 50 op te halen en de geretourneerde waarde te traceren.

```
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));
```

Gebruik de methode `getPixels()` als u niet de waarde van één pixel, maar die van een groep pixels wilt lezen. Deze methode genereert een bytearray op basis van een rechthoekig gebied met pixelgegevens dat wordt doorgegeven als een parameter. Alle elementen van de bytearray (de pixelwaarden) zijn gehele getallen zonder teken: 32-bits niet-vermenigvuldigde pixelwaarden.

Wanneer u de waarde van een groep pixels wilt instellen of wijzigen, gebruikt u de methode `setPixels()`. Deze methode verwacht twee parameters (`rect` en `inputByteArray`) die worden gecombineerd om een rechthoekig gebied (`rect`) met pixelgegevens (`inputByteArray`) te produceren.

Terwijl gegevens worden gelezen (en geschreven) uit `inputByteArray`, wordt de methode `ByteArray.readUnsignedInt()` aangeroepen voor elk van de pixels in de array. Als `inputByteArray` om welke reden dan ook niet een volledige rechthoek aan pixelgegevens bevat, worden de afbeeldingsgegevens niet meer verwerkt.

Voor het ophalen en instellen van pixelgegevens verwacht de bytearray 32-bits ARGB-pixelwaarden.

In het volgende voorbeeld worden de methoden `getPixels()` en `setPixels()` gebruikt om een groep pixels van het ene naar het andere object BitmapData te kopiëren:

```
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;
```

Botsingdetectie op pixelniveau

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `BitmapData.hitTest()` wordt botsingdetectie op pixelniveau uitgevoerd tussen de bitmapgegevens en een ander object of punt.

De methode `BitmapData.hitTest()` accepteert vijf parameters:

- `firstPoint` (Point): deze parameter verwijst naar de pixelpositie van de linkerbovenhoek van het eerste object BitmapData waarop de detectietest wordt uitgevoerd.
- `firstAlphaThreshold` (uint): met deze parameter wordt de hoogste alfakanaalwaarde opgegeven die voor deze detectietest als dekkend wordt beschouwd.
- `secondObject` (Object): deze parameter vertegenwoordigt het betrokken gebied. `secondObject` kan een object Rectangle, Point, Bitmap of BitmapData zijn. Dit object staat voor het raakgebied waarin de botsingdetectie wordt uitgevoerd.

- `secondBitmapDataPoint` (Point): deze optionele parameter wordt gebruikt om een pixellocatie in het tweede object BitmapData te definiëren. Deze parameter wordt alleen gebruikt wanneer de waarde van `secondObject` een object BitmapData is. De standaardwaarde is `null`.
- `secondAlphaThreshold` (uint): deze optionele parameter vertegenwoordigt de hoogste alfakanaalwaarde die in het tweede object BitmapData als dekkend wordt beschouwd. De standaardwaarde is 1. Deze parameter wordt alleen gebruikt wanneer de waarde van `secondObject` een object BitmapData is en beide BitmapData-objecten transparant zijn.

Wanneer u botsingdetectie uitvoert voor dekkende afbeeldingen, worden deze in ActionScript beschouwd als volledig dekkende rechthoeken (of selectiekaders). Wanneer u een detectietest op pixelniveau wilt uitvoeren voor transparante afbeeldingen, moeten beide afbeeldingen transparant zijn. Daarnaast gebruikt ActionScript de parameters voor de alfadrempel om te bepalen op welk punt de pixels niet meer transparant zijn, maar dekkend.

In het volgende voorbeeld worden drie bitmapafbeeldingen gemaakt en wordt op twee verschillende punten gecontroleerd of er pixelbotsingen optreden (met als resultaten een keer false en een keer 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
```

Bitmapgegevens kopiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voor het kopiëren van bitmapgegevens van de ene afbeelding naar de andere kunt u verschillende methoden gebruiken: `clone()`, `copyPixels()`, `copyChannel()`, `draw()` en `drawWithQuality()`. (De methode `drawWithQuality`is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.)

Zoals de naam al doet vermoeden, kunt u met de methode `clone()` bitmapgegevens uit het ene naar het andere object BitmapData klonen. Wanneer deze methode wordt aangeroepen, wordt er een nieuw object BitmapData geretourneerd dat een exacte kloon is van de oorspronkelijke instantie waaruit het object is gekopieerd.

In het volgende voorbeeld wordt een kopie van een oranje (bovenliggend) vierkant gekloond en wordt de kloon naast het oorspronkelijke bovenliggende vierkant geplaatst:

```
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;
```

Met de methode `copyPixels()` kunt u snel en eenvoudig pixels van het ene naar het andere object BitmapData kopiëren. Er wordt een rechthoekige momentopname (gedefinieerd door de parameter `sourceRect`) van de bronafbeelding gemaakt en deze wordt naar een ander rechthoekig gebied (van gelijke grootte) gekopieerd. De locatie van de nieuwe geplakte rechthoek wordt gedefinieerd in de parameter `destPoint`.

Met de methode `copyChannel()` wordt een vooraf gedefinieerde kleurkanaalwaarde (alfa, rood, groen of blauw) gesampled uit een BitmapData-bronobject en wordt deze gekopieerd naar een kanaal van een BitmapData-doelobject. Wanneer deze methode wordt aangeroepen, heeft dit geen invloed op de andere kanalen in het BitmapData-doelobject.

Met de methoden `draw()` en `drawWithQuality()` wordt de grafische inhoud uit een bronsprite, filmclip of een ander weergaveobject getekend, ofwel gerenderd, in een nieuwe bitmap. U kunt de parameters `matrix`, `colorTransform`, `blendMode` en de doelparameter `clipRect` gebruiken om de wijze waarop de nieuwe bitmap wordt gerenderd te wijzigen. Bij deze methode wordt de vectorrenderer in Flash Player en AIR gebruikt om de gegevens te genereren.

Wanneer u `draw()` of `drawWithQuality()` aanroept, geeft u het bronobject (sprite, filmclip of ander weergaveobject) door als de eerste parameter, zoals hier wordt gedemonstreerd:

```
myBitmap.draw(movieClip);
```

Als op het bronobject transformaties (kleur, matrix enzovoort) zijn toegepast sinds het is geladen, worden deze niet naar het nieuwe object gekopieerd. Als u de transformaties naar de nieuwe bitmap wilt kopiëren, moet u de waarde van de eigenschap `transform` uit het oorspronkelijke object kopiëren naar de eigenschap `transform` van het object Bitmap dat het nieuwe object BitmapData gebruikt.

Bitmapgegevens comprimeren

Flash Player 11.3 of hoger, AIR 3.3 of hoger

Met de methode `flash.display.BitmapData.encode()` worden bitmapgegevens door het besturingssysteem gecomprimeerd in een van de volgende indelingen voor afbeeldingscompressie:

- **PNG** - Maakt gebruik van PNG-compressie en optioneel van snelle compressie waarbij compressiesnelheid belangrijker is dan de bestandsgrootte. Als u PNG-compressie wilt gebruiken, geeft u een nieuw `flash.display.PNGEncoderOptions`-object door als de tweede parameter van de methode `BitmapData.encode()`.
- **JPEG** - Maakt gebruik van JPEG-compressie, waarbij u desgewenst de afbeeldingskwaliteit kunt opgeven. Als u JPEG-compressie wilt gebruiken, geeft u een nieuw `flash.display.JPENGEncoderOptions`-object door als de tweede parameter van de methode `BitmapData.encode()`.
- **JPEGXR** - Maakt gebruik van JPEG Extended Range (XR)-compressie, waarbij u desgewenst instellingen voor kleurkanaal, verlies en entropie kunt opgeven. Als u JPEGXR-compressie wilt gebruiken, geeft u een nieuw `flash.display.JPEGXREncoderOptions`-object door als de tweede parameter van de methode `BitmapData.encode()`.

U kunt deze functie voor het verwerken van afbeeldingen gebruiken in een workflow voor het uploaden naar of het downloaden van een server.

Met het volgende voorbeeldsnippet wordt een BitmapData-object gecomprimeerd aan de hand van `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);
```

Structuren met ruisfuncties maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u de weergave van een bitmap wilt wijzigen, kunt u hierop een ruiseffect toepassen met de methode `noise()` of `perlinNoise()`. Een ruiseffect kan worden vergeleken met het beeld van een nog niet afgestelde televisie.

Als u een ruiseffect op een bitmap wilt toepassen, gebruikt u de methode `noise()`. Met deze methode wordt een willekeurige kleurwaarde toegepast op de pixels in een opgegeven gebied van een bitmapafbeelding.

Deze methode accepteert vijf parameters:

- `randomSeed` (int): het willekeurige zaadgetal waarmee het patroon wordt bepaald. Ondanks de naam levert dit getal dezelfde resultaten op als wanneer hetzelfde getal wordt doorgegeven. Voor een echt willekeurig resultaat moet u de methode `Math.random()` gebruiken om een willekeurig getal door te geven voor deze parameter.
- `low` (uint): deze parameter verwijst naar de laagste waarde die voor elk kanaal (0 tot en met 255) moet worden gegenereerd. De standaardwaarde is 0. Een lagere instelling resulteert in een donkerder ruispatroon, een hogere instelling in een helderder patroon.

- `high` (uint): deze parameter verwijst naar de hoogste waarde die voor elk kanaal (0 tot en met 255) moet worden gegenereerd. De standaardwaarde is 255. Een lagere instelling resulteert in een donkerder ruispatroon, een hogere instelling in een helderder patroon.
- `channelOptions` (uint): met deze parameter wordt opgegeven op welk kleurkanaal van het bitmapobject het ruispatroon wordt toegepast. Het getal kan uit een combinatie van elk van de vier ARGB-kleurkanaalwaarden bestaan: De standaardwaarde is 7.
- `grayScale` (Boolean): bij `true` wordt de waarde `randomSeed` toegepast op de bitmappixels, zodat alle kleuren worden verwijderd uit de afbeelding. Het alfakanaal wordt niet beïnvloed door deze parameter. De standaardwaarde is `false`.

In het volgende voorbeeld wordt een bitmapafbeelding gemaakt en wordt er een blauw ruispatroon op toegepast:

```
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);
        }
    }
}
```

Als u een meer organische structuur wilt maken, gebruikt u de methode `perlinNoise()`. De methode `perlinNoise()` produceert realistische, organische structuren die ideaal zijn voor rook, wolken, water, vuur en zelfs explosies.

Omdat deze wordt gegenereerd door een algoritme, gebruikt de methode `perlinNoise()` minder geheugen dan op bitmaps gebaseerde structuren. Het kan echter nog steeds invloed hebben om processorgebruik, doordat het uw inhoud vertraagt zodat het scherm langzamer dan de framesnelheid wordt ververst, vooral op oudere computers. Dit wordt voornamelijk veroorzaakt door de drijvende-kommaberekeningen die moeten worden uitgevoerd om de Perlin-ruisalgoritmes te verwerken.

De methode accepteert negen parameters (de eerste zes zijn vereist):

- `baseX` (Number): hiermee wordt de x-waarde (grootte) van gemaakte patronen bepaald.
- `baseY` (Number): hiermee wordt de y-waarde (grootte) van gemaakte patronen bepaald.
- `numOctaves` (uint): aantal octaven of afzonderlijke ruisfuncties dat moet worden gecombineerd om deze ruis te maken. Grote aantallen octaven resulteren in gedetailleerdere afbeeldingen, maar nemen ook meer verwerkingstijd in beslag.
- `randomSeed` (int): het willekeurige zaadgetal werkt op exact dezelfde wijze als in de functie `noise()`. Voor een echt willekeurig resultaat moet u de methode `Math.random()` gebruiken om een willekeurig getal door te geven voor deze parameter.
- `stitch` (Boolean): bij `true` probeert de methode de overgangsranden van de afbeelding vloeiend te maken om een naadloze structuur te maken voor bitmapvullingstegels.

- `fractalNoise` (Boolean): deze parameter heeft betrekking op de randen van de verlopen die worden gegenereerd door de methode. Bij `true` genereert de methode fractale ruis waarmee de randen van het effect vloeiend worden gemaakt. Bij `false` wordt er turbulentie gegenereerd. Een afbeelding met turbulentie heeft zichtbare onderbrekingen in het verloop, waardoor het een betere benadering geeft van scherpere visuele effecten, zoals vlammen en golfslag.
- `channelOptions` (uint): de parameter `channelOptions` werkt op exact dezelfde wijze als bij de methode `noise()`. Met deze parameter wordt opgegeven op welk kleurkanaal (van de bitmap) het ruispatroon wordt toegepast. Het getal kan uit een combinatie van elk van de vier ARGB-kleurkanaalwaarden bestaan: De standaardwaarde is 7.
- `grayScale` (Boolean): de parameter `grayScale` werkt op exact dezelfde wijze als bij de methode `noise()`. Bij `true` wordt de waarde `randomSeed` toegepast op de bitmappixels, zodat alle kleuren worden verwijderd uit de afbeelding. De standaardwaarde is `false`.
- `offsets` (Array): een array van punten die overeenkomen met x- en y-verschuivingen voor elke octaaf. Door de verschuivingswaarden te bewerken, kunt u de lagen van de afbeelding vloeiend verschuiven. Elk punt in de verschuivingsarray heeft invloed op een specifieke octaafruisfunctie. De standaardwaarde is `null`.

In het volgende voorbeeld wordt een object BitmapData van 150 x 150 pixels gemaakt dat de methode `perlinNoise()` aanroept om een groen en blauw wolkeneffect te genereren:

```
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);
        }
    }
}
```

Schuiven in bitmaps

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Stel dat u een toepassing voor straatkaarten hebt gemaakt waarbij u de weergave telkens moet bijwerken wanneer de gebruiker de kaart verplaatst (ook al zijn het maar een paar pixels).

U kunt deze functionaliteit maken door in te stellen dat elke keer dat de gebruiker de kaart verplaatst, een nieuwe afbeelding met de bijgewerkte kaartweergave moet worden gerenderd. U kunt ook één grote afbeelding maken met behulp van de `scroll()`-methode.

Met de methode `scroll()` wordt het op het scherm weergegeven gedeelte van de bitmap gekopieerd en vervolgens verplaatst naar de verschoven locatie die is opgegeven met de parameters (`x`, `y`). Als een gedeelte van de bitmap zich buiten het werkgebied bevindt, levert dit een verplaatsingseffect op. Gecombineerd met een timerfunctie (of de gebeurtenis `enterFrame`) kunt u een animatie- of schuifeffect genereren.

In het volgende voorbeeld wordt van de Perlin-ruis uit het vorige voorbeeld een grotere bitmapafbeelding gegenereerd (waarvan driekwart buiten het werkgebied wordt gerenderd). Vervolgens wordt de methode `scroll()` toegepast in combinatie met de gebeurtenislistener `enterFrame` die de afbeelding met één pixel diagonaal omlaag verplaatst. Deze methode wordt telkens aangeroepen wanneer het frame wordt geopend. Het resultaat is dat gedeelten van de afbeelding die niet op het scherm worden weergegeven wel in het werkgebied verschijnen wanneer de afbeelding naar beneden wordt verschoven.

```
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);
}
```

Gebruikmaken van mipmapping

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

MIP-maps (ook wel *mipmaps* genoemd) zijn gegroepeerde bitmaps die zijn gekoppeld aan een structuur om de kwaliteit en de prestaties van het renderen in runtime te verhogen. Elke bitmapafbeelding in de mipmap is een versie van de hoofdbitmapafbeelding, maar bevat minder details dan de hoofdafbeelding.

U kunt bijvoorbeeld een mipmap hebben met een hoofdafbeelding van de hoogste kwaliteit van 64x64 pixels. De afbeeldingen van lagere kwaliteit in de mipmap hebben dan een resolutie van 32x32, 16x16, 8x8, 4x4, 2x2 of 1x1 pixels.

Structuurstreaming verwijst naar de mogelijkheid eerst een bitmap van lage kwaliteit te laden en vervolgens bitmaps van steeds hogere kwaliteit weer te geven naarmate deze worden geladen. Aangezien bitmaps van lagere kwaliteit klein zijn, worden deze sneller geladen dan de hoofdafbeelding. Gebruikers kunnen dan een afbeelding in een toepassing bekijken voordat de hoofdafbeelding van hoge kwaliteit is geladen.

In Flash Player 9.115.0. en hoger en AIR wordt deze technologie geïmplementeerd (het proces wordt *mipmapping* genoemd) door geoptimaliseerde versies te maken van variërende schalen van elke bitmap (te beginnen bij 50%).

Flash Player 11.3 en AIR 3.3 bieden ondersteuning voor structuurstreaming via de parameter `streamingLevels` van de methoden `Context3D.createCubeTexture()` en `Context3D.createTexture()`.

Met structuurcompressie kunt u structuurafbeeldingen rechtstreeks op de GPU opslaan in een gecomprimeerde indeling om GPU-geheugen en geheugenbandbreedte te sparen. Gecomprimeerde structuren worden gewoonlijk offline gecomprimeerd en geüpload naar de GPU in een gecomprimeerde indeling. Flash Player 11.4 en AIR 3.4 ondersteunen echter runtimestructuurcompressie. Dit is vooral handig in bepaalde situaties, zoals bij het renderen van dynamische structuren van vectorafbeeldingen. Voer de volgende stappen uit als u runtimestructuurcompressie wilt gebruiken:

- Maak het structuurobject door de `Context3D.createTexture()`-methode aan te roepen of door `flash.display3D.Context3DTextureFormat.COMPRESSED` of `flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA` door te geven in de derde parameter.
- Gebruik de `flash.display3D.textures.Texture`-instantie die wordt geretourneerd door `createTexture()` en roep `flash.display3D.textures.Texture.uploadFromBitmapData()` of `flash.display3D.textures.Texture.uploadFromByteArray()` aan. Met deze methoden kunt u de structuur in één stap uploaden en comprimeren.

MIP-mappen worden gemaakt voor de volgende types bitmaps:

- een bitmap (JPEG, GIF of PNG-bestanden) die wordt weergegeven met de Loader-klasse van ActionScript 3.0.
- een bitmap in de bibliotheek van een Flash Professional-document
- een BitmapData-object
- een bitmap die wordt weergegeven met behulp van de functie `loadMovie()` van ActionScript 2.0.

MIP-maps worden niet toegepast bij gefilterde objecten op filmfragmenten die in bitmaps in de cache zijn geplaatst. MIP-maps worden wel toegepast als er bitmaptransformaties in een gefilterd weergaveobject zijn, zelfs als de bitmap zich in de gemaskeerde inhoud bevindt.

Mipmapping gebeurt automatisch, maar u kunt een aantal richtlijnen volgen, zodat uw afbeeldingen gebruik maken van deze optimalisatie:

- Voor het afspelen van video stelt u de eigenschap `smoothing` in op `true` voor het Video-object (zie de klasse Video
- Voor bitmaps hoeft u de eigenschap `smoothing` niet in te stellen op `true`, maar de kwaliteitsverbeteringen zijn beter zichtbaar als dit wel wordt gedaan.
- Gebruik voor tweedimensionale afbeeldingen bitmapafmetingen die deelbaar zijn door 4 of 8 (bijvoorbeeld 640x128, een waarde die vervolgens als volgt kan worden verminderd: 320x64 > 160x32 > 80x16 > 40x8 > 20x4 > 10x2 > 5x1).

  Gebruik voor driedimensionale structuren mipmaps waarbij elke afbeelding een resolutie tot de macht 2 heeft (dus 2^n). Stel bijvoorbeeld dat de hoofdafbeelding een resolutie van 1024x1024 pixels heeft. De kwaliteit van de lagere afbeeldingen in de mipmap zou dan 512x512, 256x256, 128x128, enz. zijn tot aan 1x1. En de mipmap zou in totaal 11 afbeeldingen bevatten.

  Mipmapping vindt niet plaats voor bitmapinhoud met een oneven breedte of hoogte.

Bitmapvoorbeeld: geanimeerde draaiende maan

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het voorbeeld van een geanimeerde draaiende maan demonstreert technieken voor het werken met objecten Bitmap en afbeeldingsgegevens van een bitmap (objecten BitmapData). In het voorbeeld wordt een animatie gemaakt van een draaiende, bolvormige maan, waarbij gebruik wordt gemaakt van een platte afbeelding van het oppervlak van de maan als onbewerkte afbeeldingsgegevens. De volgende technieken worden getoond:

- Het laden van een externe afbeelding en het werken met de onbewerkte afbeeldingsgegevens ervan
- Het maken van animatie door herhaaldelijk pixels te kopiëren van andere delen van een bronafbeelding
- Bitmapafbeelding maken door pixelwaarden in te stellen

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de toepassingsbestanden voor de geanimeerde draaiende maan in de map Samples/SpinningMoon. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
SpinningMoon.mxml of SpinningMoon.fla	Het hoofdtoepassingsbestand in Flex (MXML) of Flash (FLA).
com/example/programmingas3/moon/MoonSphere.as	Klasse die de functionaliteit uitvoert van het laden, weergeven en bewegen van de maan.
moonMap.png	Afbeeldingsbestand met een foto van het oppervlak van de maan, die wordt geladen en gebruikt om de geanimeerde draaiende maan te maken.

Externe afbeelding als bitmapgegevens laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eerste hoofdtaak die u uitvoert, is het laden van een extern afbeeldingsbestand: een foto van het oppervlak van de maan. De laadbewerking wordt afgehandeld door twee methoden in de klasse MoonSphere: de constructor `MoonSphere()`, waar het laadproces wordt gestart, en de methode `imageLoadComplete()`, die wordt aangeroepen als de externe afbeelding volledig is geladen.

Het laden van een externe afbeelding komt overeen met het laden van een extern SWF-bestand. Beide gebruiken een instantie van de klasse flash.display.Loader voor het uitvoeren van de laadbewerking. De werkelijke code van de methode `MoonSphere()` waarmee het laden van de afbeelding wordt gestart, is als volgt:

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

De eerste regel declareert de instantie Loader met de naam `imageLoader`. De derde regel begint daadwerkelijk het laadproces door de methode `load()` van het object Loader aan te roepen, waarbij een instantie URLRequest wordt doorgegeven die de URL vertegenwoordigt van de afbeelding die moet worden geladen. De tweede regel stelt de gebeurtenislistener in die wordt geactiveerd wanneer de afbeelding volledig is geladen. De methode `addEventListener()` wordt niet aangeroepen voor de instantie Loader zelf, maar aangeroepen voor de eigenschap `contentLoaderInfo` van het object Loader. De instantie Loader verzendt zelf geen gebeurtenissen met betrekking tot

de inhoud die wordt geladen. De eigenschap `contentLoaderInfo` ervan bevat echter een verwijzing naar het object LoaderInfo dat is gekoppeld aan de inhoud die wordt geladen in het object Loader (in dit geval de externe afbeelding). Dat object LoaderInfo biedt wel verschillende gebeurtenissen met betrekking tot de voortgang en voltooiing van het laden van de externe inhoud, waaronder de gebeurtenis `complete` (`Event.COMPLETE`) die een aanroep van de methode `imageLoadComplete()` activeert wanneer de afbeelding volledig is geladen.

Hoewel het starten van het laden van de externe afbeelding een belangrijk onderdeel van het proces vormt, is het net zo belangrijk om te weten wat moet worden gedaan nadat het laden is voltooid. Zoals getoond in de code hierboven, wordt de functie `imageLoadComplete()` aangeroepen wanneer de afbeelding is geladen. Die functie doet verschillende dingen met geladen afbeeldingsgegevens, die hieronder worden beschreven. De afbeeldingsgegevens kunnen echter alleen worden gebruikt wanneer ze kunnen worden benaderd. Wanneer een object Loader wordt gebruikt om een externe afbeelding te laden, wordt de geladen afbeelding een instantie Bitmap die is gekoppeld als een onderliggend weergaveobject van het object Loader. In dit geval is de instantie Loader beschikbaar voor de gebeurtenislistenermethode als onderdeel van het gebeurtenisobject dat als parameter aan de methode wordt doorgegeven. De eerste regels van de methode `imageLoadComplete()` zijn als volgt:

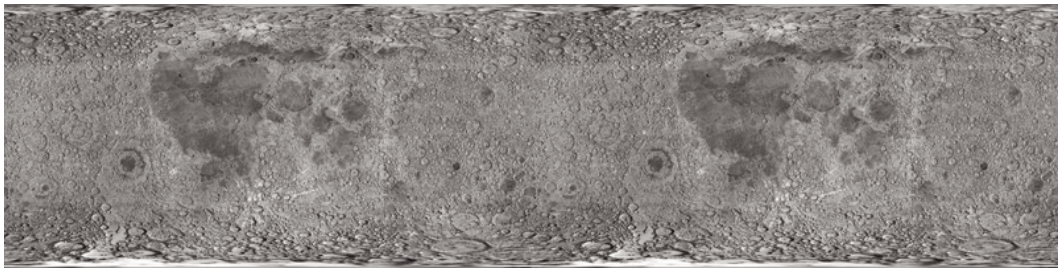
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

De gebeurtenisobjectparameter heeft de naam `event` en is een instantie van de klasse Event. Elke instantie van de klasse Event heeft een eigenschap `target` die verwijst naar het object dat de gebeurtenis activeert (in dit geval de instantie LoaderInfo waarvoor de methode `addEventListener()` is aangeroepen, zoals hierboven beschreven). Het object LoaderInfo heeft op zijn beurt een eigenschap `content` die (zodra het laadproces is voltooid) de instantie Bitmap bevat met de geladen bitmapafbeelding. Wanneer u de afbeelding direct op het scherm wilt weergegeven, kunt u deze instantie Bitmap (`event.target.content`) koppelen aan een weergaveobjectcontainer. (U kunt het object Loader ook aan een weergaveobjectcontainer koppelen.) In dit voorbeeld wordt de geladen inhoud echter gebruikt als bron van onbewerkte afbeeldingsgegevens in plaats van dat deze op het scherm wordt weergegeven. De eerste regel van de `imageLoadComplete()`-methode leest de `bitmapData`-eigenschap van de geladen Bitmap-instantie (`event.target.content.bitmapData`) en slaat deze op in de instantievariabele `textureMap`, die als bron van de afbeeldingsgegevens wordt gebruikt om de animatie van de roterende maan te maken. Dit wordt hieronder beschreven.

Animatie maken door pixels te kopiëren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een basisdefinitie van een animatie is de illusie van beweging, of verandering, die wordt veroorzaakt door een afbeelding gedurende een tijdsperiode te veranderen. In dit voorbeeld is het doel het maken van het beeld van een bolvormige maan die rond zijn verticale as draait. Voor deze animatie kunt u het bolvormige vervormingsaspect echter negeren. Bekijk de werkelijke afbeelding die wordt geladen en gebruikt als de bron van de maanafbeeldingsgegevens:



Zoals u kunt zien, bestaat deze afbeelding niet uit een of meer bollen, maar is deze een rechthoekige foto van het oppervlak van de maan. Omdat de foto precies bij de evenaar van de maan is genomen, zijn de delen van de afbeelding die zich dichter bij de boven- en onderkant bevinden, uitgerekt en vervormd. Voor het verwijderen van deze vervorming uit de afbeelding en het bolvormiger maken van de afbeelding, gebruikt u een verschuivingskaartfilter, zoals later wordt beschreven. Omdat deze bronafbeelding rechthoekig is, moet de code de foto van het maanoppervlak gewoon horizontaal bewegen om de illusie te creëren, dat de bol draait.

De afbeelding bevat eigenlijk twee exemplaren van de foto van het maanoppervlak naast elkaar. Deze afbeelding is de bronafbeelding waaruit afbeeldingsgegevens herhaaldelijk worden gekopieerd om de illusie van beweging te maken. Wanneer twee exemplaren van de afbeelding naast elkaar liggen, is het eenvoudiger een doorlopend, ononderbroken schuifeffect te maken. Het animatieproces wordt hieronder stap voor stap beschreven.

Het proces bestaat eigenlijk uit twee afzonderlijke ActionScript-objecten. Het eerste object is de geladen bronafbeelding, die in de code wordt vertegenwoordigd door de instantie BitmapData met de naam `textureMap`. Zoals eerder beschreven, wordt `textureMap` gevuld met afbeeldingsgegevens zodra de externe afbeelding is geladen. Hiervoor wordt de volgende code gebruikt:

```
textureMap = event.target.content.bitmapData;
```

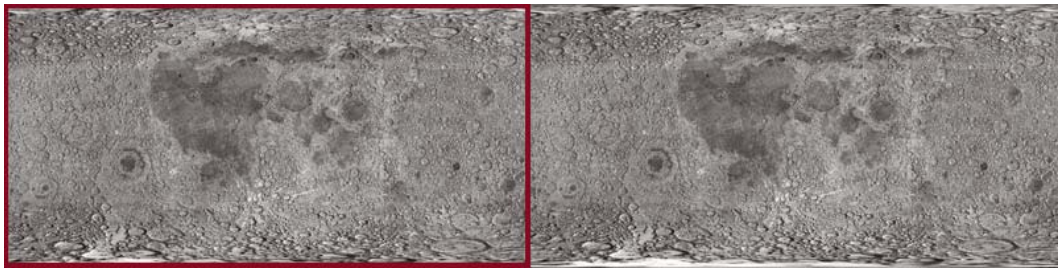
De inhoud van `textureMap` is de rechthoekige afbeelding van de maan. Om een geanimeerde rotatie te maken, gebruikt de code een Bitmapinstantie`sphere`, dit is het werkelijke weergaveobject dat de maan op het scherm weergeeft. Op dezelfde manier als bij `textureMap` wordt het object `sphere` gemaakt en gevuld met de oorspronkelijke afbeeldingsgegevens uit de methode `imageLoadComplete()`, waarbij de volgende code wordt gebruikt:

```
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));
```

Zoals de code laat zien, wordt `sphere` geïnstantieerd. De eigenschap `bitmapData` (de onbewerkte afbeeldingsgegevens die door `sphere` worden weergegeven) wordt gemaakt met dezelfde hoogte en half de breedte van `textureMap`. Met andere woorden, de inhoud van `sphere` krijgt de grootte van één maanfoto (aangezien de afbeelding `textureMap` twee maanfoto's naast elkaar bevat). Vervolgens wordt de eigenschap `bitmapData` gevuld met afbeeldingsgegevens via de methode `copyPixels()`. De parameters in de aanroep van de methode `copyPixels()` geven verschillende handelingen aan:

- De eerste parameter geeft aan dat de afbeeldingsgegevens worden gekopieerd van `textureMap`.
- De tweede parameter, een nieuwe instantie Rectangle, geeft aan van welk deel van de `textureMap` de momentopname moet worden genomen. In dit geval is de momentopname een rechthoek die begint in de linkerbovenhoek van `textureMap` (zoals aangegeven door de eerste twee parameters `Rectangle(): 0, 0`) en komen de breedte en de hoogte van de momentopname overeen met de eigenschappen `width` en `height` van `sphere`.
- De derde parameter, een nieuwe instantie Point met de x- en y-waarden `0`, definieert de bestemming van de pixelgegevens, in dit geval de linkerbovenhoek (0, 0) van `sphere.bitmapData`.

Zoals zichtbaar weergegeven in de afbeelding hieronder, kopieert de code de pixels van `textureMap` en plakt deze op `sphere`. Met andere woorden, de BitmapData-inhoud van `sphere` is het hieronder gemarkeerde deel van `textureMap`:



Dit is echter alleen nog maar de eerste status van `sphere` (de eerste afbeeldingsinhoud die op `sphere` wordt gekopieerd).

Nu de bronafbeelding is geladen en `sphere` is gemaakt, bestaat de laatste taak van de methode `imageLoadComplete()` uit het instellen van de animatie. De animatie werkt op basis van een instantie Timer met de naam `rotationTimer`, die wordt gemaakt en geactiveerd met de volgende code:

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

De code maakt eerst een instantie Timer met de naam `rotationTimer`. De parameter die is doorgegeven aan de constructor `Timer()` geeft aan dat `rotationTimer` de gebeurtenis `timer` elke 15 milliseconden moet activeren. Vervolgens wordt de methode `addEventListener()` aangeroepen en wordt opgegeven dat wanneer de gebeurtenis `timer` optreedt (`TimerEvent.TIMER`), de methode `rotateMoon()` wordt aangeroepen. Ten slotte wordt de timer daadwerkelijk gestart door de methode `start()` ervan aan te roepen.

Door de manier waarop `rotationTimer` is gedefinieerd, roept Flash Player de methode `rotateMoon()` in de klasse MoonSphere ongeveer elke 15 milliseconden aan, waardoor beweging van de maan plaatsvindt. De broncode van de `rotateMoon()`-methode is als volgt:

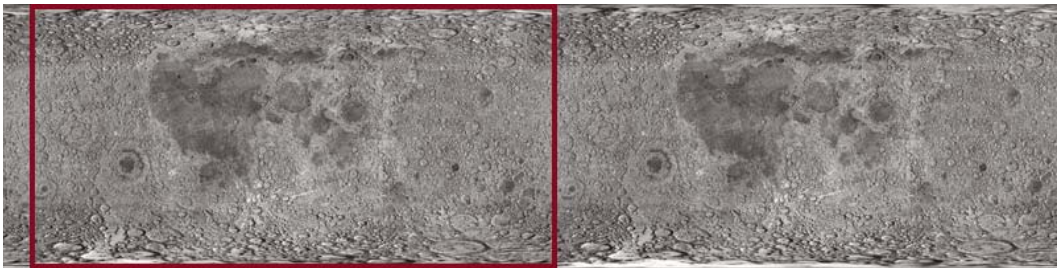
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                    new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                    new Point(0, 0));

    event.updateAfterEvent();
}
```

In deze code gebeuren drie dingen:

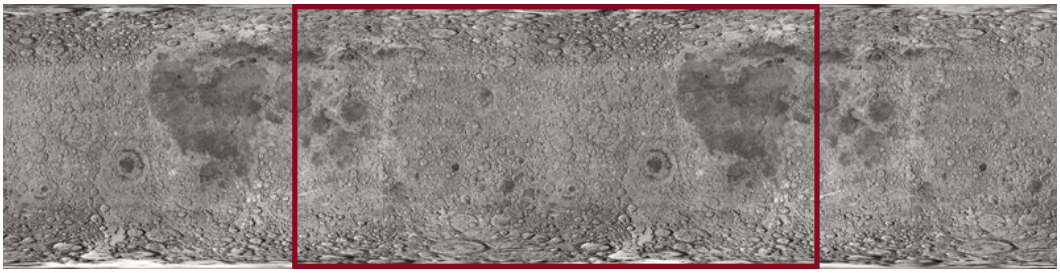
1 De waarde van de variabele `sourceX` (oorspronkelijk op 0 ingesteld) neemt met 1 toe.

```
sourceX += 1;
```

Zoals u zult zien, wordt `sourceX` gebruikt om de locatie te bepalen in `textureMap` waarvan de pixels worden gekopieerd op `sphere`, zodat met deze code het effect wordt bereikt dat de rechthoek met één pixel naar rechts op `textureMap` beweegt. Zoals weergegeven in de afbeelding, is de bronrechthoek na enkele animatiecycli met verschillende pixels naar rechts verplaatst:

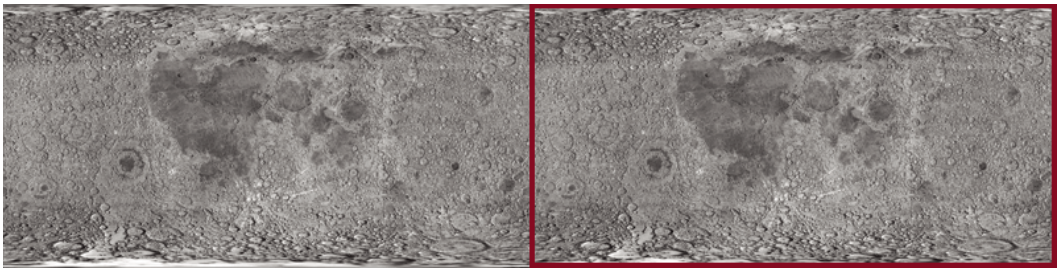


Na verloop van meer animatiecycli is de rechthoek nog verder verplaatst:



Deze geleidelijke, evenwichtige verschuiving van de locatie waarvan de pixels worden gekopieerd, vormt de sleutel tot animatie. Door de bronlocatie langzaam en continu naar rechts te verplaatsen, lijkt het alsof de afbeelding die op het scherm wordt weergegeven in `sphere` continu naar links schuift. Dit is de reden waarom de bronafbeelding (`textureMap`) twee exemplaren nodig heeft van de foto van het maanoppervlak. Omdat de rechthoek continu naar rechts beweegt, bevindt deze zich het merendeel van de tijd niet boven één enkele maanfoto worden de twee maanfoto's overlapt.

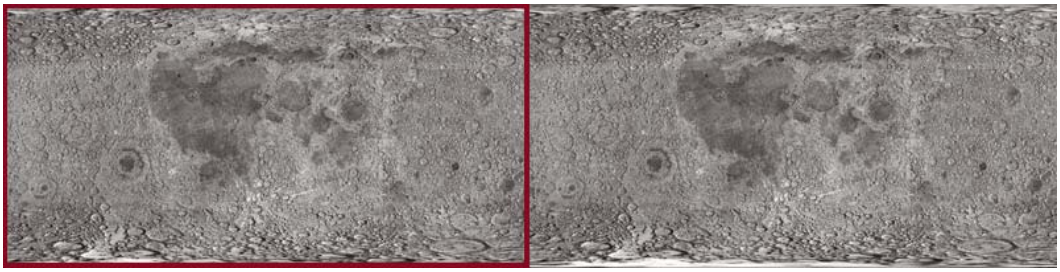
2 Er kleeft één probleem aan de langzaam naar rechts bewegende bronrechthoek. Na verloop van tijd bereikt de rechthoek de rechterrand van `textureMap` en zijn er geen maanfotopixels meer over om op `sphere` te kopiëren:



Met de volgende coderegels wordt dit probleem opgelost:

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

De code controleert of `sourceX` (de linkerrand van de rechthoek) het midden van `textureMap` heeft bereikt. Zo ja, dan herstelt de code de waarde van `sourceX` naar 0, waardoor de rechthoek weer naar de linkerrand van `textureMap` wordt verplaatst en de cyclus opnieuw wordt gestart:



3 Nu de juiste waarde van `sourceX` is berekend, bestaat de laatste stap van het maken van de animatie uit het daadwerkelijk kopiëren van de nieuwe bronrechthoekpixels op `sphere`. De code waarmee dit wordt gedaan, lijkt sterk op de code die `sphere` eerder vulde (eerder beschreven); het enige verschil is dat in dit geval, in de constructoraanroep `new Rectangle()`, de linkerrand van de rechthoek op `sourceX` wordt geplaatst:

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

Deze code wordt herhaaldelijk aangeroepen, elke 15 milliseconden. Omdat de locatie van de bronrechthoek continu verschuift en de pixels op `sphere` worden gekopieerd, lijkt het op het scherm alsof de maanfoto die wordt vertegenwoordigd door `sphere` voortdurend schuift. Met andere woorden, het lijkt alsof de maan voortdurend draait.

Bolvormig uiterlijk maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De maan is uiteraard een bol en geen rechthoek. Daarom moet het voorbeeld de voortdurend draaiende rechthoekige foto van het maanoppervlak in een bol omzetten. Dit proces omvat twee afzonderlijke stappen: het gebruik van een masker om alle inhoud te verbergen behalve een cirkelvormig gebied van de foto van het maanoppervlak en het gebruik van een verschuivingskaartfilter om het uiterlijk van de maanfoto te vervormen en het driedimensionaal te laten lijken.

Eerst wordt een cirkelvormig masker gebruikt om alle inhoud van het object MoonSphere te verbergen, met uitzondering van de bol die door het filter is gemaakt. De volgende code maakt het masker als een instantie Shape en past deze toe als het masker van de instantie MoonSphere:

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

Omdat MoonSphere een weergaveobject is (het is gebaseerd op de klasse Sprite), kan het masker direct op de instantie MoonSphere wordt toegepast met gebruik van de overgeërfde eigenschap `mask`.

Het alleen verbergen van delen van de foto via een cirkelvormig masker is niet genoeg om een realistische draaiende bol te maken. Vanwege de manier waarop de foto van het maanoppervlak is genomen, zijn de afmetingen niet proportioneel; de delen van de afbeelding die zich bij de boven- of onderkant van de afbeelding bevinden, zijn meer vervormd en uitgerekt in vergelijking met de delen van de evenaar. Er wordt een verschuivingskaartfilter gebruikt om de weergave van de maanfoto te vervormen om deze meer driedimensionaal te laten lijken.

Een verschuivingskaartfilter is een type filter dat wordt gebruikt om een afbeelding te vervormen. In dit geval wordt de maanfoto 'vervormd' om de foto realistischer te maken. De boven- en onderkant van de afbeelding worden hiertoe horizontaal samengedrukt, terwijl het midden ongewijzigd blijft. Vooropgesteld dat het filter werkt op een vierkant deel van de foto, zorgt het samendrukken van de boven- en onderkant ervoor dat het vierkant wordt veranderd in een cirkel. Een bijeffect van het bewegen van deze vervormde afbeelding is dat het midden van de afbeelding met meer pixels lijkt te worden verplaatst dan de gebieden die zich dicht bij de boven- en onderkant bevinden; dit zorgt voor de illusie dat de cirkel een driedimensionaal object is (een bol).

De volgende code wordt gebruikt om het verschuivingskaartfilter `displaceFilter` te maken:

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

De eerste parameter, `fisheyeLens`, staat bekend als de kaartafbeelding; in dit geval is het een object BitmapData dat programmatisch wordt gemaakt. De creatie van die afbeelding wordt beschreven in "Bitmapafbeelding maken door pixelwaarden in te stellen" op pagina 276. De overige parameters beschrijven de positie in de gefilterde afbeelding waarop het filter moet worden toegepast, welke kleurkanalen moeten worden gebruikt om het verschuivingseffect te beheren en in welke mate deze de verschuiving beïnvloeden. Zodra het verschuivingskaartfilter is gemaakt, wordt dit toegepast op `sphere`, nog steeds binnen de methode `imageLoadComplete()`:

```
sphere.filters = [displaceFilter];
```

De uiteindelijke afbeelding, met toegepast masker en verschuivingskaartfilter, ziet er als volgt uit:

Met elke cyclus van de draaiende maananimatie wordt de BitmapData-inhoud van sphere overschreven door een nieuwe momentopname van de bronafbeeldingsgegevens. Het filter hoeft echter niet telkens opnieuw te worden toegepast. Dit komt omdat het filter wordt toegepast op de instantie Bitmap (het weergaveobject) in plaats van op de bitmapgegevens (de onbewerkte pixelinformatie). De instantie Bitmap bevat niet de daadwerkelijke bitmapgegevens, maar is een weergaveobject dat de bitmapgegevens op het scherm weergeeft. Om het als een analogie te beschrijven: een Bitmap-instantie is als een diaprojector die wordt gebruikt om fotodia's op een scherm weer te geven en een BitmapData-object is als de fotodia die met behulp van een diaprojector kan worden weergegeven. Er kan direct een filter op een object BitmapData worden toegepast. Dit kan worden vergeleken met het direct tekenen op een fotografische dia om de afbeelding te veranderen. Een filter kan ook op om het even welk weergaveobject worden toegepast, inclusief een instantie Bitmap. Dit kan worden vergeleken met het plaatsen van een filter voor de lens van de diaprojector om de uitvoer op het scherm te vervormen (zonder de oorspronkelijke dia te wijzigen). Omdat de onbewerkte bitmapgegevens benaderbaar zijn via de eigenschap bitmapData van de instantie Bitmap, zou het filter direct kunnen zijn toegepast op de onbewerkte bitmapgegevens. In dit geval is het echter zinniger het filter op het weergaveobject Bitmap toe te passen in plaats van op de bitmapgegevens.

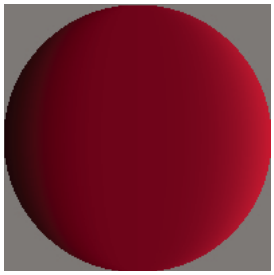
Zie “Weergaveobjecten filteren” op pagina 283 voor meer informatie over het gebruik van het verschuivingskaartfilter in ActionScript.

Bitmapafbeelding maken door pixelwaarden in te stellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een belangrijk aspect van een verschuivingskaartfilter is dat het twee afbeeldingen nodig heeft. Eén afbeelding, de bronafbeelding, is de afbeelding die wordt gewijzigd door het filter. In dit voorbeeld is de bronafbeelding de instantie Bitmap met de naam `sphere`. De andere afbeelding die door het filter wordt gebruikt, is de kaartafbeelding. De kaartafbeelding wordt niet op het scherm weergegeven. In plaats daarvan wordt de kleur van elke pixel ervan gebruikt als invoer voor de verschuivingsfunctie; de kleur van de pixel op een bepaalde x-, y-coördinaat in de kaartafbeelding bepaalt hoeveel verschuiving (fysieke positieverschuiving) wordt toegepast op de pixel op die x-, y-coördinaat in de bronafbeelding.

Als gevolg hiervan heeft het voorbeeld de juiste kaartafbeelding nodig om het verschuivingskaartfilter te kunnen gebruiken voor het creëren van een bolvormig effect: een afbeelding met een grijze achtergrond en een cirkel die is gevuld met een enkele verlopende kleur (rood) die horizontaal van donker naar licht gaat, zoals hieronder wordt getoond:



Omdat slechts één kaartafbeelding en filter worden gebruikt in dit voorbeeld, wordt de kaartafbeelding slechts eenmaal gemaakt in de methode `imageLoadComplete()` (met andere woorden wanneer de externe afbeelding is geladen). De kaartafbeelding, `fisheyeLens`, wordt gemaakt door de methode `createFisheyeMap()` van de klasse MoonSphere aan te roepen:

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

Binnen de methode `createFisheyeMap()` wordt de kaartafbeelding met één pixel per keer getekend via de methode `setPixel()` van de klasse BitmapData. De volledige code voor de methode `createFisheyeMap()` wordt hieronder weergegeven, gevolgd door een stapsgewijze beschrijving van de werking ervan:

```
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;
}
```

Op het moment dat de methode wordt aangeroepen, ontvangt deze een parameter, `radius`, die de straal aangeeft van de cirkelvormige afbeelding die moet worden gemaakt. Vervolgens maakt de code het object BitmapData waarop de cirkel wordt getekend. Dit object, `result`, wordt uiteindelijk doorgegeven als de geretourneerde waarde van de methode. Zoals wordt getoond in het volgende codefragment, wordt de BitmapData-instantie `result` gemaakt met een breedte en hoogte die net zo groot zijn als de diameter van de cirkel, zonder transparantie (`false` voor de derde parameter) en vooraf gevuld met de kleur `0x808080` (normaal grijs):

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

Vervolgens gebruikt de code twee lussen om elke pixel van de afbeelding te doorlopen. De buitenste lus doorloopt elke kolom in de afbeelding van links naar rechts (de variabele `i` wordt gebruikt om de horizontale positie te vertegenwoordigen van de pixel die momenteel wordt gemanipuleerd), terwijl de binnenste lus elke pixel in de huidige kolom van boven naar beneden doorloopt (de variabele `j` vertegenwoordigt de verticale positie van de huidige pixel). De code voor de lussen (de inhoud van de binnenste lus is weggelaten) wordt hieronder getoond:

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

Terwijl de lussen de pixels één voor één doorlopen, wordt voor elke pixel een waarde (de kleurwaarde van die pixel in de kleurafbeelding) berekend. Dit proces omvat vier stappen:

1 De code berekent de afstand van de huidige pixel tot het midden van de cirkel langs de x-as (`i - straal`). Die waarde wordt gedeeld door de straal om het een percentage van de straal te maken in plaats van een absolute afstand (`(i - straal) / straal`). Deze percentagewaarde wordt opgeslagen in een variabele met de naam `pctX` en de equivalente waarde voor de y-as wordt berekend en opgeslagen in de variabele `pctY`, zoals in de volgende code wordt getoond:

```
var pctX:Number = (i - radius) / radius;
var pctY:Number = (j - radius) / radius;
```

2 Met gebruikmaking van een trigonometrische standaardformule, de stelling van Pythagoras, wordt de lineaire afstand berekend tussen het midden van de cirkel en het huidige punt op basis van `pctX` en `pctY`. Die waarde wordt in een variabele met de naam `pctDistance` opgeslagen, zoals hieronder wordt getoond:

```
var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
```

3 Vervolgens controleert de code of het afstandspercentage minde dan 1 is (of 100% van de radius, met andere woorden als de pixel zich binnen de cirkelradius bevindt). Wanneer de pixel zich binnen de cirkel bevindt, krijgt deze een berekende kleurwaarde toegewezen (hier weggelaten, maar dit wordt later in stap 4 beschreven) en anders gebeurt er verder niets en wordt de pixelkleur niet gewijzigd, zodat de pixel normaalgrijs (standaard) blijft:

```
if (pctDistance < 1)
{
    ...
}
```

4 Voor de pixels die zich binnen de cirkel bevinden, wordt een kleurwaarde berekend. De uiteindelijke kleur is een roodtint tussen zwart (0% rood) aan de linkerrand van de cirkel en helderrood (100%) aan de rechterrand van de cirkel. De kleurwaarde wordt oorspronkelijk berekend in drie delen (rood, groen en blauw), zoals hieronder wordt getoond:

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

Alleen het rode deel van de kleur (de variabele `red`) heeft een waarde. De groene en blauwe waarden (de variabelen `green` en `blue`) worden hier ter verduidelijking getoond, maar kunnen worden weggelaten. Omdat het doel van deze methode het maken van een cirkel met een rood verloop is, zijn geen groene of blauwe waarden nodig.

Zodra de drie afzonderlijke kleurwaarden zijn bepaald, worden deze tot een enkele kleurwaarde (geheel getal) gecombineerd via een bitschuivend standaardalgoritme, zoals in de volgende code wordt getoond:

```
rgb = (red << 16 | green << 8 | blue);
```

De berekende kleurwaarde wordt uiteindelijk toegewezen aan de huidige pixel met de methode `setPixel()` van het BitmapData-object `result`, zoals hieronder wordt getoond.

```
result.setPixel(i, j, rgb);
```

Bitmapafbeeldingen asynchroon decoderen

Flash Player 11 of hoger, Adobe AIR 2.6 of hoger

Wanneer u met bitmapafbeeldingen werkt, kunt u deze op asynchrone wijze decoderen en laden om de prestaties van uw toepassing te verbeteren. In veel gevallen duurt het asynchroon decoderen van een bitmapafbeelding even lang als het synchroon decoderen. De bitmapafbeelding wordt echter gedecodeerd in een aparte thread voordat het verwante `Loader`-object de gebeurtenis `COMPLETE` verzendt. U kunt grotere afbeeldingen dus asynchroon decoderen nadat u deze hebt geladen.

Met de klasse `ImageDecodingPolicy` in het pakket `flash.system` kunt u het schema voor het laden van de bitmap opgeven. Standaard worden bitmapafbeeldingen synchroon geladen.

Beleid voor het decoderen van bitmaps	Schema voor het laden van bitmaps	Beschrijving
`ImageDecodingPolicy.ON_DEMAND`	Synchroon	Geladen afbeeldingen worden gedecodeerd wanneer de afbeeldingsgegevens worden benaderd. Gebruik dit beleid om kleinere afbeeldingen te decoderen. U kunt dit beleid ook gebruiken wanneer uw toepassing niet afhankelijk is van ingewikkelde effecten en overgangen.
`ImageDecodingPolicy.ON_LOAD`	Asynchroon	Geladen afbeeldingen worden tijdens het laden gedecodeerd, voordat de gebeurtenis `COMPLETE` wordt verzonden. Ideaal voor grotere afbeeldingen (groter dan 10 MP). Wanneer u op AIR gebaseerde mobiele toepassingen met paginaovergangen ontwikkelt, gebruikt u dit beleid voor het laden van bitmaps, zodat uw toepassing beter lijkt te presteren.

Opmerking: *Als het bestand dat wordt geladen een bitmapafbeelding is en het decoderingsbeleid `ON_LOAD` wordt gebruikt, wordt de afbeelding asynchroon gedecodeerd voordat de gebeurtenis `COMPLETE` wordt verzonden.*

De volgende code illustreert het gebruik van de klasse `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);
```

U kunt `ON_DEMAND`-decodering nog steeds gebruiken met de methoden `Loader.load()` en `Loader.loadBytes()`. Alle andere methoden die een `LoaderContext`-object als een argument gebruiken, negeren alle doorgegeven `ImageDecodingPolicy`-waarden.

Het volgende voorbeeld illustreert het verschil tussen het asynchroon en synchroon decoderen van een bitmapafbeelding:

```
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");
        }
    }
}
```

Zie Thibaud Imbert: Asynchronous bitmap decoding in the Adobe Flash runtimes voor een demo van de gevolgen van verschillende vormen van decoderingsbeleid.

Hoofdstuk 14: Weergaveobjecten filteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing van filtereffecten op bitmapafbeeldingen is altijd het domein geweest van gespecialiseerde beeldbewerkingssoftware, zoals Adobe Photoshop® en Adobe Fireworks®. ActionScript 3.0 bevat het flash.filters-pakket, dat een aantal filterklassen van bitmapeffecten bevat. Met deze effecten kunnen ontwikkelaars programmatisch filters toepassen op bitmaps en weergaveobjecten en veel van de effecten bereiken die beschikbaar zijn in beeldbewerkingsgtoepassingen.

Basisbeginselen van het filteren van weergaveobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een toepassing onder andere verfijnen door het toevoegen van eenvoudige grafische effecten. U kunt een slagschaduw achter een foto plaatsen om een 3D-effect te krijgen of een gloed rondom een knop om te tonen dat deze actief is. ActionScript 3.0 bevat tien filters die u kunt toepassen op elk weergaveobject of op een instantie BitmapData. Deze variëren van basisfilters, zoals de slagschaduw- en gloedfilters, tot complexe filters zoals het verschuivingskaartfilter en het convolutiefilter.

Opmerking: *Naast de ingebouwde filters, kunt u ook aangepaste filters en effecten programmeren met Pixel Bender. Zie* *"Werken met Pixel Bender-arceringen" op pagina 318.*

Belangrijke concepten en termen

De volgende lijst bevat belangrijke termen die u bij het maken van filters zult tegenkomen:

Schuine kant Een rand die wordt gevormd door het lichter maken van pixels aan twee zijden en het donkerder maken van pixels aan de twee tegenovergestelde zijden. Met dit effect creëert u een driedimensionale rand. Het effect wordt gebruikt voor verheven of dieper liggende knoppen en vergelijkbare afbeeldingen.

Convolutie Pixels in een afbeelding vervormen door elke waarde van de pixel te combineren met de waarden van een aantal of alle naastgelegen pixels, waarbij gebruik wordt gemaakt van verschillende verhoudingen.

Verschuiving Pixels in een afbeelding verschuiven of verplaatsen naar een nieuwe positie.

Matrix Een raster van getallen dat wordt gebruikt voor het uitvoeren van bepaalde wiskundige berekeningen door de getallen in het raster toe te passen op diverse waarden en vervolgens de resultaten te combineren.

Meer Help-onderwerpen

flash.filters-pakket

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

Filters maken en toepassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met filters kunt u een reeks effecten toepassen op bitmap- en weergaveobjecten die variëren van slagschaduwen tot schuine kanten en vervagingen. Elk filter wordt gedefinieerd als een klasse. Bij het toepassen van filters worden dan ook instanties van filterobjecten gemaakt, net als bij het maken van elk ander object. Wanneer u een instantie van een filterobject hebt gemaakt, kan deze eenvoudig worden toegepast op een weergaveobject met de eigenschap `filters` van het object of, in het geval van een object BitmapData, met de methode `applyFilter()`.

Een filter maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Om een filterobject te maken, roept u gewoon de constructormethode van de door u geselecteerde filterklasse aan. Om bijvoorbeeld een DropShadowFilter-object te maken, gebruikt u de volgende code:

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

Hoewel ze hier niet worden getoond, accepteert de constructor `DropShadowFilter()` (net als alle constructors van filterklassen) diverse optionele parameters die kunnen worden gebruikt om de weergave van het filtereffect aan te passen.

Een filter toepassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een filterobject hebt gemaakt, kunt u deze toepassen op een weergaveobject of een object BitmapData. De wijze waarop u het filter toepast, hangt af van het object waarop het filter wordt toegepast.

Een filter toepassen op een weergaveobject

Wanneer u filtereffecten toepast op een weergaveobject, doet u dit via de eigenschap `filters`. De eigenschap `filters` van een weergaveobject is een instantie Array waarvan de elementen de filterobjecten zijn die worden toegepast op het weergaveobject. Wanneer u een enkel filter op een weergaveobject wilt toepassen, maakt u de filterinstantie, voegt u deze toe aan een instantie Array en wijst u dat object Array aan de eigenschap `filters` van het weergaveobject toe:

```
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;
```

Als u meerdere filters wilt toewijzen aan het object, kunt u alle filters toevoegen aan de instantie Array voordat u deze toewijst aan de eigenschap `filters`. U kunt meerdere objecten toevoegen aan een array door deze als parameters door te geven aan de constructor. Deze code past bijvoorbeeld een schuine-kantfilter en een gloedfilter toe op het eerder gemaakte weergaveobject.

```
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;
```

Wanneer u de array met de filters maakt, kunt u deze maken met behulp van de constructor `new Array()` (zoals wordt getoond in de voorgaande voorbeelden) of u kunt gebruikmaken van de literale Array-syntaxis door de filters tussen vierkante haakjes (`[]`) te zetten. Deze coderegel bijvoorbeeld:

```
var filters:Array = new Array(dropShadow, blur);
```

doet hetzelfde als deze coderegel:

```
var filters:Array = [dropShadow, blur];
```

Wanneer u meerdere filters toepast op weergaveobjecten, worden deze op een cumulatieve, opeenvolgende wijze toegepast. Wanneer een array filters bijvoorbeeld twee elementen bevat, waarbij een schuine-kantfilter als eerste en een slagschaduwfilter als tweede is toegevoegd, wordt het slagschaduwfilter toegepast op zowel het schuine-kantfilter als het weergaveobject. Dit komt doordat het slagschaduwfilter de tweede positie in de array filters inneemt. Als u filter op een niet-cumulatieve manier wilt toepassen, past u elke filter toe op een nieuwe kopie van het weergaveobject.

Wanneer u slechts één filter of een paar filters toewijst aan een weergaveobject, kunt u een filterinstantie maken en deze in een enkele instructie toewijzen aan het object. De volgende coderegel past bijvoorbeeld een vervagend filter toe op een weergaveobject `myDisplayObject`.

```
myDisplayObject.filters = [new BlurFilter()];
```

De vorige code maakt een Array-instantie met een letterlijke Array-syntaxis (vierkante haakjes), een BlurFilter-instantie als een element in Array en wijst die Array toe aan de eigenschap `filters` van het weergaveobject `myDisplayObject`.

Filters verwijderen uit een weergaveobject

Het verwijderen van alle filters uit een weergaveobject is net zo eenvoudig als het toewijzen van een waarde null aan de eigenschap `filters`:

```
myDisplayObject.filters = null;
```

Wanneer u meerdere filters hebt toegepast op een object en u wilt slechts een van de filters verwijderen, moet u diverse stappen doorlopen om de array met eigenschap `filters` te wijzigen. Zie "Potentiële problemen bij het werken met filters" op pagina 286 voor meer informatie.

Een filter toepassen op een object BitmapData

Wanneer u een filter wilt toepassen op een object BitmapData, moet u gebruikmaken van de methode `applyFilter()` van het object BitmapData:

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

De methode `applyFilter()` past een filter toe op een bronobject BitmapData en produceert een nieuwe, gefilterde afbeelding. Deze methode wijzigt niet de oorspronkelijke bronafbeelding. In plaats daarvan wordt het resultaat van het filter dat wordt toegepast op de bronafbeelding, opgeslagen in de instantie BitmapData waarvoor de methode `applyFilter()` wordt aangeroepen.

De werking van filters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij het filteren van weergaveobjecten wordt een kopie van het oorspronkelijke object als een transparante bitmap in cache geplaatst.

Wanneer een filter op een weergaveobject is toegepast, slaat de runtime het object op als een bitmap, zolang het object een geldige filterlijst heeft. Deze bronbitmap wordt vervolgens gebruikt als de oorspronkelijke afbeelding voor alle volgende toegepaste filtereffecten.

Weergaveobjecten bevatten meestal twee bitmaps: een bitmap met het oorspronkelijk ongefilterde bronweergaveobject en een bitmap voor de uiteindelijke afbeelding na filtering. De uiteindelijke afbeelding wordt voor weergave gebruikt. Zolang het weergaveobject niet wordt gewijzigd, hoeft de uiteindelijke afbeelding niet te worden bijgewerkt.

Potentiële problemen bij het werken met filters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn diverse potentiële oorzaken die kunnen leiden tot verwarring of problemen wanneer u werkt met filters.

Filters en bitmap in cache plaatsen

Wanneer u een filter wilt toepassen op een weergaveobject, moet het in cache plaatsen van een bitmap zijn ingeschakeld voor dat object. Wanneer u een filter op een weergaveobject toepast, waarvan de eigenschap `cacheAsBitmap` is ingesteld op `false`, wordt de eigenschap `cacheAsBitmap` van het object automatisch ingesteld op `true`. Als u later alle filters van het weergaveobject verwijdert, wordt de eigenschap `cacheAsBitmap` teruggezet naar de vorige waarde.

Filters wijzigen bij uitvoering

Als er op een weergaveobject al een of meer filters zijn toegepast, kunt u de set filters niet wijzigen door extra filters toe te voegen of door filters te verwijderen uit de array met eigenschappen `filters`. In plaats daarvan moet u, om wijzigingen aan te brengen in de set filters die wordt toegepast, de gewenste wijzigingen doorvoeren in een afzonderlijke array, waarna u deze array toewijst aan de array met eigenschappen filter van het weergaveobject voor de filters die op het object worden toegepast. De eenvoudigste manier om dit te doen is door de array met eigenschappen `filters` in een willekeurige variabele Array in te lezen en uw wijzigingen in deze tijdelijke array aan te brengen. Vervolgens wijst u deze array weer toe aan de eigenschap `filters` van het weergaveobject. In complexere gevallen kan het zijn dat u een afzonderlijke hoofdarray met filters moet maken. U brengt de wijzigingen aan in de hoofdarray met filters en na elke wijziging wijst u de hoofdarray toe aan de eigenschap `filters` van het weergaveobject.

Extra filters toevoegen

In de volgende code wordt geïllustreerd hoe u een extra filter toevoegt aan een weergaveobject waarop al een of meer filters zijn toegepast. Eerst wordt een gloedfilter toegepast op het weergaveobject met de naam `myDisplayObject`. Wanneer later op het weergaveobject wordt geklikt, wordt de functie `addFilters()` aangeroepen. In deze functie worden twee aanvullende filters toegepast op `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);
```

Eén filter uit een set filters verwijderen

Als er meerdere filters op een weergaveobject zijn toegepast en u wilt een van deze filters verwijderen (waarbij de andere filters op het object toegepast blijven), gaat u als volgt te werk: kopieer de filters naar een tijdelijke array, verwijder het ongewenste filter uit deze array en wijs de tijdelijke array toe aan de eigenschap `filters` van het weergaveobject. In "Waarden ophalen en arrayelementen verwijderen" op pagina 33 worden verschillende manieren beschreven waarop u een of meer elementen uit een array kunt verwijderen.

De meest voor de hand liggende situatie is het verwijderen van het bovenste filter van het object (het laatste filter dat op het object is toegepast). Gebruik de methode `pop()` van de klasse Array om het filter uit de array te verwijderen:

```
// 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;
```

Om het onderste filter te verwijderen (het eerste filter dat op het object is toegepast) gebruikt u dezelfde code, waarbij u de methode `pop()` van de klasse Array vervangt door de methode `shift()`.

Als u een filter in het midden van een array met filters wilt verwijderen (indien de array meer dan twee filters bevat), kunt u de methode `splice()` gebruiken. U moet weten wat de index (de positie in de array) is van het filter dat u wilt verwijderen. Met de volgende code verwijdert u bijvoorbeeld het tweede filter (het filter op index 1) van een weergaveobject:

```
// 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;
```

De index van een filter bepalen

U moet weten welk filter u uit de array wilt verwijderen, dus u moet weten wat de index van dat filter is. Het kan zijn dat de index van het filter dat u wilt verwijderen bekend is (als gevolg van de manier waarop de toepassing is ontworpen), anders moet u deze berekenen.

Het is het handigste om de toepassing zo te ontwerpen dat het filter dat u wilt verwijderen altijd dezelfde positie inneemt in de set filters. Stel dat u één weergaveobject hebt waarop een convolutiefilter en een slagschaduwfilter zijn toegepast (in die volgorde) en u wilt het slagschaduwfilter verwijderen maar het convolutiefilter behouden. U weet dan op welke positie het filter zich bevindt (het bovenste filter), zodat u meteen weet welke Array-methode u moet gebruiken (in dit geval `Array.pop()`) om het slagschaduwfilter te verwijderen.

Als het filter dat u wilt verwijderen altijd een bepaald type is, maar zich niet altijd op dezelfde positie in de set filters bevindt, kunt u het gegevenstype van elk filter in de array vaststellen om te bepalen welk filter u moet verwijderen. Met de volgende code kunt u bijvoorbeeld bepalen welk filter in een set filter een gloedfilter is en dit filter vervolgens uit de set verwijderen.

```
// 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;
```

In complexere gevallen, bijvoorbeeld als het filter dat verwijderd moet worden tijdens runtime moet worden geselecteerd, kunt u het beste een afzonderlijk en onveranderlijk exemplaar van de filterarray opslaan als de hoofdlijst met filters. Wanneer u een wijziging aanbrengt in de set filters, wijzigt u de hoofdlijst en past u vervolgens de filterarray toe als de eigenschap `filters` van het weergaveobject.

In de volgende code worden bijvoorbeeld meerdere convolutiefilters toegepast op een weergaveobject om verschillende visuele effecten aan te brengen. Later wordt een van deze filters verwijderd, terwijl de andere behouden blijven. In dit geval wordt een hoofdarray met filters en een verwijzing naar het filter dat verwijderd moet worden opgeslagen. Het zoeken naar en het verwijderen van het specifieke filter verloopt hetzelfde als de hiervoor beschreven procedure, alleen wordt er geen tijdelijk exemplaar van de array met filters gemaakt, maar de hoofdarray wordt bewerkt en vervolgens op het weergaveobject toegepast.

```
// 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;
}
```

In deze procedure (wanneer u een opgeslagen filterverwijzing vergelijkt met de items in de array met filters om te bepalen welk filter verwijderd moet worden) *moet* u een afzonderlijk exemplaar van de array met filters opslaan. De code werkt niet als u de opgeslagen filterverwijzing vergelijkt met de elementen in de tijdelijke array die is gekopieerd van de eigenschap `filters` van het weergaveobject. Dit gebeurt, omdat de runtime intern een kopie maakt van elk filterobject in de array, wanneer u een array aan de eigenschap `filters` toewijst. Deze kopieën (en dus niet de oorspronkelijke objecten) worden op het weergaveobject toegepast. Wanneer u vervolgens de eigenschap `filters` in de tijdelijke array inleest, bevat de tijdelijke array verwijzingen naar de gekopieerde filterobjecten in plaats van verwijzingen naar de oorspronkelijke filterobjecten. Als u vervolgens in het voorgaande voorbeeld probeert te bepalen wat de index van `filterToRemove` is door dit te vergelijken met de filters in een tijdelijke array met filters, wordt er geen overeenkomst gevonden.

Filters en objecttransformaties

Geen enkel gefilterd gebied (bijvoorbeeld slagschaduw) buiten het begrenzende vak van een weergaveobject wordt beschouwd als een onderdeel van het oppervlak voor raakdetectiedoeleinden (bepalen of een instantie een andere instantie overlapt of doorsnijdt). Omdat raakdetectie is gebaseerd op vectoren, kunt u geen raakdetectie uitvoeren op het bitmapresultaat. Wanneer u bijvoorbeeld een schuine-kantfilter toepast op een knopinstantie, is de raakdetectie niet beschikbaar op het schuine-kantdeel van de instantie.

Schalen, roteren en scheeftrekken worden niet ondersteund door filters. Wanneer het gefilterde weergaveobject zelf is geschaald (wanneer `scaleX` en `scaleY` niet 100% zijn), schaalt het filtereffect niet met de instantie mee. Dit betekent dat de oorspronkelijke vorm van de instantie roteert, schaalt of scheeftrekt, maar het filter roteert of schaalt niet en trekt niet scheef met de instantie.

U kunt een instantie laten bewegen met een filter om realistische effecten te maken of instanties nesten en de klasse BitmapData gebruiken om filters te laten bewegen om dit effect te bereiken.

Filters en objecten Bitmap

Wanneer u een filter toepast op een object BitmapData, wordt de eigenschap `cacheAsBitmap` automatisch ingesteld op `true`. Op deze manier wordt het filter feitelijk toegepast op de kopie van het object in plaats van op het origineel.

Deze kopie wordt vervolgens zo dicht mogelijk bij de dichtstbijzijnde pixel op de hoofdweergave (over het oorspronkelijke object) geplaatst. Wanneer de grenzen van de bitmap worden gewijzigd, wordt de gefilterde kopie van de bitmap opnieuw op basis van het origineel gemaakt en dus niet uitgerekt of vervormd.

Wanneer u alle filters voor een weergaveobject wist, wordt de eigenschap `cacheAsBitmap` opnieuw ingesteld op de waarde waarop deze was ingesteld voor de toepassing van het filter.

Beschikbare weergavefilters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 bevat tien filterklassen die u kunt toepassen op weergave- en BitmapData-objecten:

- Schuine-kantfilter (klasse BevelFilter)
- Vervagende filter (klasse BlurFilter)
- Slagschaduwfilter (klasse DropShadowFilter)
- Gloedfilter (klasse GlowFilter)
- Verlopende schuine-kantfilter (klasse GradientBevelFilter)
- Verlopende-gloedfilter (klasse GradientGlowFilter)
- Kleurmatrixfilter (klasse ColorMatrixFilter)
- Convolutiefilter (klasse ConvolutionFilter)
- Verschuivingskaartfilter (klasse DisplacementMapFilter)
- Arcering, filter (ShaderFilter, klasse)

De eerste zes filters zijn eenvoudige filters die kunnen worden gebruikt voor het toepassen van een specifiek effect, waarbij een aantal aanpassingen van het effect mogelijk zijn. Deze zes filters kunnen worden toegepast met behulp van ActionScript. Ze kunnen ook op objecten in Flash Professional worden toegepast via het deelvenster Filters. Zelfs wanneer u filters toepast met behulp van ActionScript, kunt u hierdoor met Flash Professional de visuele interface snel verschillende filters en instellingen uitproberen en uitzoeken hoe u een gewenst effect kunt creëren.

De laatste vier filters zijn alleen beschikbaar in ActionScript. Deze filters, het kleurmatrixfilter, convolutiefilter, verschuivingskaartfilter en arceringsfilter, zijn veel flexibeler wat betreft het soort effecten dat met deze filters gemaakt kan worden. Deze filters zijn niet geoptimaliseerd voor één effect, maar bieden kracht en flexibiliteit. Door verschillende waarden te selecteren voor de matrix, kan de convolutiefilter bijvoorbeeld worden gebruikt voor het toepassen van effecten als vervagen, reliëf, verscherpen, kleurgrenzen zoeken, transformaties en nog veel meer.

Elk van de filters, of ze nu eenvoudig of complex zijn, kan worden aangepast met behulp van de eigenschappen. In het algemeen hebt u twee keuzen voor het instellen van filtereigenschappen. Bij alle filters kunt u de eigenschappen instellen door de parameterwaarden door te geven aan de constructor van het filterobject. U kunt daarentegen de filters later aanpassen door waarden voor de eigenschappen van het filterobject in te stellen, of u nu wel of niet de filtereigenschappen hebt ingesteld via parameters. De meeste voorbeeldcodelijsten stellen eigenschappen direct in om het voorbeeld begrijpelijker te maken. U kunt echter meestal hetzelfde resultaat bereiken met minder coderegels wanneer u de waarden als parameters doorgeeft in de constructor van het filterobject. Zie de vermeldingen voor het pakket flash.filters in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de details van elk filter en de bijbehorende eigenschappen en constructorparameters.

Schuine-kantfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse BevelFilter kunt u een 3D-rand met schuine kant aan het gefilterde object toevoegen. Door dit filter lijkt het alsof de scherpe hoeken of kanten van het object zijn weggebeiteld of afgekant.

Met de eigenschappen van de klasse BevelFilter kunt u de weergave van de schuine kant aanpassen. U kunt markeer- en schaduwkleuren instellen evenals vervagingen, hoeken en plaatsing van schuine kanten. U kunt zelfs een uitneemeffect toepassen.

In het volgende voorbeeld wordt een externe afbeelding geladen en wordt er een schuine-kantfilter op toegepast.

```
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];
```

Vervagend filter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse BlurFilter kunt u een weergaveobject inclusief inhoud uitsmeren of wazig maken. Vervagende effecten zijn handig om de indruk te geven dat een object onscherp is of om snelle bewegingen te simuleren, zoals in een bewegingsvervaging. Door de eigenschap `quality` van het vervagende filter in te stellen op laag, kunt u een licht onscherpe-lenseffect simuleren. Wanneer u de eigenschap `quality` instelt op hoog, krijgt u een vloeiend vervagend effect dat lijkt op Gaussiaans vervagen.

In het volgende voorbeeld wordt een cirkelobject gemaakt met de methode `drawCircle()` van de klasse Graphics en wordt er een vervagende filter op toegepast:

```
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];
```

Slagschaduwfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Slagschaduwen geven de indruk dat er een afzonderlijke lichtbron is boven het doelobject. De positie en intensiteit van deze lichtbron kunnen worden gewijzigd om een verscheidenheid aan verschillende slagschaduweffecten te produceren.

De klasse DropShadowFilter gebruikt een algoritme die gelijkt op de algoritme voor het filter Wazig maken. Het voornaamste verschil is dat het slagschaduwfilter een aantal eigenschappen meer heeft die u kunt wijzigen om verschillende kenmerken van lichtbronnen te simuleren (zoals alfa, kleur, verschuiving en helderheid).

Met het slagschaduwfilter kunt u bovendien aangepaste transformatieopties toepassen op de stijl van de slagschaduw, waaronder binnen- of buitenschaduw en uitnemenmodus.

In de volgende code wordt een vierkant sprite-vak gemaakt en wordt er een slagschaduwfilter op toegepast.

```
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];
```

Gloedfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse GlowFilter past u een lichteffect op weergaveobjecten toe, zodat ze worden weergegeven alsof er vanachter het object met een lamp op wordt geschenen, zodat een zachte gloed ontstaat.

Net als bij het slagschaduwfilter bevat het gloedfilter eigenschappen waarmee de afstand, de hoek en de kleur van de lichtbron kunnen worden gewijzigd voor het produceren van variërende effecten. Het GlowFilter heeft verschillende opties voor het wijzigen van de stijl van de gloed, waaronder binnen- of buitengloed en uitnemenmodus.

In de volgende code wordt een kruis gemaakt met behulp van de klasse Sprite en wordt er een gloedfilter op toegepast.

```
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];
```

Verlopende schuine-kantfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse GradientBevelFilter kunt u een uitgebreid schuinekant-effect toepassen op weergave- of BitmapData-objecten. Wanneer u een verloopkleur toepast op de schuine kant, wordt de ruimtelijke diepte van de schuine kant verbeterd waardoor de kanten een realistischere 3D-weergave krijgen.

In de volgende code wordt een rechthoekig object gemaakt met behulp van de methode `drawRect()` van de klasse Shape en wordt er een verlopende schuine-kantfilter op toegepast.

```
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);
```

Verlopende-gloedfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse GradientGlowFilter kunt u een uitgebreid gloedeffect toepassen op weergave- of BitmapData-objecten. Het effect biedt meer controle over de kleur van de gloed zodat een realistischer gloedeffect kan worden geproduceerd. Bovendien kunt u met het verlopende-gloedfilter een verlopende gloed toepassen op de binnen-, buiten- of bovenranden van een object.

In het volgende voorbeeld wordt een cirkel getekend in het werkgebied waarop een verlopende-gloedfilter wordt toegepast. Wanneer u de muis verder naar rechts en omlaag beweegt, neemt de mate van vervaging toe in respectievelijk horizontale en verticale richtingen. Elke keer wanneer u in het werkgebied klikt, neemt bovendien de sterkte van de vervaging toe.

```
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);
```

Voorbeeld: Basisfilters combineren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het volgende codevoorbeeld worden verschillende basisfilters gebruikt in combinatie met een timer voor herhalende handelingen, voor het toepassen van een geanimeerde stoplichtsimulatie.

```
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();
```

Kleurmatrixfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse ColorMatrixFilter wordt gebruikt om de kleur- en alfawaarden van het gefilterde object te manipuleren. U kunt nu veranderingen in verzadiging, kleurrotatie (), luminantie bij alfa en andere effecten van kleurmanipulatie toepassen door waarden van een kleurkanaal te gebruiken en deze mogelijkerwijs toe te passen op andere kanalen.

In dat geval doorloopt het filter de pixels in de bronafbeelding een voor een en scheidt deze elke pixel in rode, groene, blauwe en alfacomponenten. Vervolgens vermenigvuldigt het filter waarden die in de kleurmatrix zijn opgegeven, met elk van deze waarden en worden de resultaten bij elkaar opgeteld om de resulterende kleurwaarde vast te stellen die wordt weergegeven op het scherm voor die pixel. De eigenschap `matrix` van het filter is een array van 20 getallen die worden gebruikt voor de berekening van de definitieve kleur. Zie het gedeelte waarin de eigenschap `matrix` van de klasse ColorMatrixFilter wordt beschreven in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over het specifieke algoritme waarmee de kleurwaarden worden berekend.

Convolutiefilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse ConvolutionFilter kunt u allerhande afbeeldingstransformaties toepassen op weergave- of BitmapData-objecten, zoals wazig maken, randdetectie, scherper maken, stansen en schuine rand.

Het convolutiefilter doorloopt een voor een alle pixels in de bronafbeelding en stelt de definitieve kleur van elke pixel vast aan de hand van de waarde van de pixel en de omringende pixels. Een matrix, die is opgegeven als een array van numerieke waarden, geeft aan in welke mate de waarde van elke afzonderlijke naastgelegen pixel de uiteindelijk resulterende waarde beïnvloedt.

Bekijk het type matrix dat het meest wordt gebruikt. Dit is een drie bij drie-matrix. De matrix bevat negen waarden:

```
N   N   N
N   P   N
N   N   N
```

Wanneer de convolutiefilter wordt toepast op een bepaalde pixel, wordt zowel naar de kleurwaarde van de pixel zelf (in het voorbeeld 'P') als naar de waarden van de omringende pixels (in het voorbeeld aangeduid als 'N') gekeken. U kunt echter waarden in de matrix instellen om op te geven hoeveel prioriteit bepaalde pixels hebben bij het beïnvloeden van de resulterende afbeelding.

De volgende matrix die is gemaakt met een convolutiefilter, laat bijvoorbeeld een afbeelding precies zoals deze was:

```
0   0   0
0   1   0
0   0   0
```

De afbeelding blijft ongewijzigd omdat de oorspronkelijke pixelwaarde een relatieve sterkte heeft van 1 bij het vaststellen van de definitieve pixelkleur, terwijl de omringende pixelwaarden een relatieve sterkte van 0 hebben (wat betekent dat de kleuren geen invloed hebben op de uiteindelijke afbeelding).

Op dezelfde manier zorgt deze matrix ervoor dat de pixels van een afbeelding een pixel naar links verschuiven:

```
0   0   0
0   0   1
0   0   0
```

In dit geval heeft de pixel zelf geen effect op de definitieve waarde van de pixel die wordt weergegeven op die locatie in de uiteindelijke afbeelding. Alleen de waarde van de pixel rechts ervan wordt gebruikt voor het vaststellen van de resulterende waarde van de pixel.

In ActionScript kunt u de matrix toepassen als een combinatie van een instantie Array met de waarden en twee eigenschappen die het aantal rijen en kolommen in de matrix opgeven. In het volgende voorbeeld wordt een afbeelding geladen en wordt na het laden een convolutiefilter toegepast op de afbeelding met behulp van de matrix uit het vorige voorbeeld:

```
// 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);
```

Wat niet direct voor de hand ligt in deze code, is het effect van het gebruik van andere waarden in de matrix dan 1 of 0. Dezelfde matrix met bijvoorbeeld het getal 8 in plaats van 1 in de positie rechts, voert dezelfde handeling uit (pixels verschuiven naar links). De kleuren van de afbeelding worden ook beïnvloed en worden 8 keer lichter weergegeven. Dit komt doordat de definitieve pixelkleurwaarden worden berekend door vermenigvuldiging van de matrixwaarden met de oorspronkelijke pixelkleuren, waarbij de waarden bij elkaar worden opgeteld en worden gedeeld door de waarde van de eigenschap `divisor` van het filter. In het codevoorbeeld is de eigenschap `divisor` ingesteld op 1. Wanneer u wilt dat de helderheid van de kleuren gelijk blijft aan de helderheid in de oorspronkelijke afbeelding, moet u ervoor zorgen dat de deler gelijk is aan de som van de matrixwaarden. Bij een matrix met een totale waarde van 8 en een deler van 1 is de resulterende afbeelding ruwweg 8 keer lichter dan de oorspronkelijke afbeelding.

Hoewel het effect van deze matrix niet direct opvalt, kunnen andere matrixwaarden worden gebruikt voor het toepassen van verschillende effecten. Hieronder volgen diverse standaardsets van matrixwaarden voor verschillende effecten bij het gebruik van een drie bij drie-matrix:

- Standaard vervagen (deler 5):

```
0 1 0
1 1 1
0 1 0
```

- Verscherpen (deler 1):

```
 0, -1, 0
-1, 5, -1
 0, -1, 0
```

- Randdetectie (deler 1):

```
 0, -1, 0
-1, 4, -1
 0, -1, 0
```

- Reliëfeffect (deler 1):

```
-2, -1, 0
-1, 1, 1
 0, 1, 2
```

  Bij de meeste van deze effecten is de deler 1. Dit komt doordat de negatieve matrixwaarden opgeteld bij de positieve matrixwaarden resulteren in 1 (of 0 in het geval van randdetectie, de waarde van de eigenschap `divisor` kan echter niet 0 zijn).

Verschuivingskaartfilter

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse DisplacementMapFilter gebruikt pixelwaarden uit een BitmapData-object (de displacement map image genoemd) om een verplaatsingseffect toe te passen op een nieuw object. De verschuivingskaartafbeelding verschilt doorgaans van het daadwerkelijke weergaveobject of de instantie BitmapData waarop het filter wordt toegepast. Bij verschuivingseffect worden pixels in de gefilterde afbeelding verschoven: met andere woorden, ze worden enigszins bij de oorspronkelijk locatie vandaan vergeplaatst. U kunt dit filter gebruiken voor het toepassen van een verschoven, kromgetrokken of gespikkeld effect.

De locatie en de mate van verschuiving die wordt toegepast op een bepaalde pixel, worden vastgelegd door de kleurwaarde van de verschuivingskaartafbeelding. Wanneer u werkt met het filter, geeft u naast de kaartafbeelding de volgende waarden op om te bepalen hoe de verschuiving wordt berekend uit de kaartafbeelding:

- Kaartpunt: De locatie op de gefilterde afbeelding waarop de linkerbovenhoek van het verschuivingsfilter wordt toegepast. U kunt dit alleen gebruiken wanneer u het filter wilt toepassen op een deel van een afbeelding.
- X-component: welk kleurkanaal van de kaartafbeelding de x-positie van pixels beïnvloedt.
- Y-component: welk kleurkanaal van de kaartafbeelding de y-positie van pixels beïnvloedt.
- X-schaal: een vermenigvuldigingswaarde die opgeeft hoe sterk de verschuiving van de x-as is.
- Y-schaal: een vermenigvuldigingswaarde die opgeeft hoe sterk de verschuiving van de y-as is.
- Filtermodus: hiermee wordt bepaald wat te doen met lege ruimten die het resultaat zijn van verplaatste pixels. De opties, die zijn gedefinieerd als constanten in de klasse DisplacementMapFilterMode, zijn: de oorspronkelijke pixels weergeven (filtermodus `IGNORE`), de pixels laten doorlopen vanaf de andere kant van de afbeelding (de standaardinstelling `WRAP`), de dichtstbijzijnde verplaatste pixel gebruiken (filtermodus `CLAMP`) of de ruimten te vullen met een kleur (filtermodus `COLOR`).

Hier volgt een eenvoudig voorbeeld ter illustratie van hoe het verplaatsingsfilter werkt. In de volgende code wordt een afbeelding geladen, wordt deze na het laden gecentreerd in het werkgebied en wordt er een verschuivingskaartfilter op toegepast waardoor de pixels in de hele afbeelding horizontaal naar links worden verschoven.

```
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);
```

De eigenschappen die worden gebruikt om de verschuiving te definiëren zijn:

- Kaartbitmap: de verschuivingsbitmap is een nieuwe instantie BitmapData die is gemaakt door de code. De afmetingen komen overeen met de afmetingen van de geladen afbeelding (de verschuiving is dus toegepast op de hele afbeelding). De bitmap wordt gevuld met effen rode pixels.
- Kaartpunt: deze waarde is ingesteld op het punt 0, 0 (ook hier wordt de verschuiving toegepast op de hele afbeelding).
- X-component: deze waarde is ingesteld op de constante `BitmapDataChannel.RED`. Dit houdt in dat de rode waarde van de kaartbitmap bepaalt in welke mate de pixels worden verschoven (worden verplaatst) langs de x-as.
- X-schaal: Deze waarde is ingesteld op 250. Bij een volledige mate van verschuiving (waarbij de kaartafbeelding volledig rood is) wordt de afbeelding slechts een klein stukje verschoven (ruwweg een halve pixel), dus wanneer deze waarde is ingesteld op 1, verschuift de afbeelding slechts 5 pixels horizontaal. Bij de instelling van 250 verschuift de afbeelding ongeveer 125 pixels.

Met deze instellingen worden pixels van de gefilterde afbeelding 250 pixels naar links verschoven. De richting (links of rechts) en mate van verschuiving worden gebaseerd op de kleurwaarde van de pixels in de kaartafbeelding. De filter doorloopt de pixels van de gefilterde afbeelding een voor een (tenminste, de pixels in het gebied waar het filter is toegepast, in dit geval alle pixels) en doet het volgende met elke pixel:

1. Flash Player vindt de overeenkomende pixel in de kaartafbeelding. Wanneer de filter bijvoorbeeld de mate van verschuiving berekent voor de pixel in de linkerbovenhoek van de gefilterde afbeelding, zoekt de filter naar de pixel in de linkerbovenhoek van de kaartafbeelding.
2. Flash Player stelt de waarde vast van het opgegeven kleurkanaal in de kaartpixel. In dit geval is de x-component van het kleurkanaal het rode kanaal en zoekt de filter de waarde van het rode kanaal van de kaartafbeelding op bij de desbetreffende pixel. Omdat de kaartafbeelding effen rood is, is het rode kanaal van de pixel 0xFF of 255. Dit wordt gebruikt als de verschuivingswaarde.
3. Flash Player vergelijkt de verschuivingswaarde met de 'middelste' waarde (127 is halverwege 0 en 255). Wanneer de verschuivingswaarde lager is dan de middelste waarde, verschuift de pixel in een positieve richting (naar rechts voor x-verschuiving, omlaag voor y-verschuiving). Wanneer de verschuivingswaarde daarentegen hoger is dan de middelste waarde (zoals in dit voorbeeld), verschuift de pixel in negatieve richting (naar links voor x-verschuiving, omhoog voor y-verschuiving). Om precies te zijn trekt de filter de verschuivingswaarde af van 127 en het resultaat (positief of negatief) is de relatieve mate van verschuiving die wordt toegepast.
4. Ten slotte bepaalt Flash Player de daadwerkelijke mate van verschuiving door vast te stellen welk percentage van volledige verschuiving de relatieve verschuivingswaarde vertegenwoordigt. In dit geval betekent volledig rood 100% verschuiving. Dat percentage wordt vervolgens vermenigvuldigd met de waarde van de x- of de y-schaal om het aantal pixels verschuiving dat wordt toegepast, vast te stellen. In dit voorbeeld bepaalt 100% keer een vermenigvuldiger van 250 de mate van verschuiving (ruwweg 125 pixels naar links).

Omdat er geen waarden zijn opgegeven voor y-component en y-schaal, zijn de standaardwaarden (die geen verschuiving veroorzaken) gebruikt. Daarom verschuift de afbeelding helemaal niet in verticale richting.

In dit voorbeeld is de standaardinstelling voor filtermodus `WRAP` gebruikt, zodat, wanneer de pixels naar links verschuiven, de lege ruimte die rechts wordt achtergelaten, wordt opgevuld met de pixels die aan de linkerkant van de afbeelding zijn verdwenen. U kunt met deze waarde experimenteren om te zien wat de verschillende effecten zijn. Wanneer u bijvoorbeeld de volgende regel toevoegt aan het deel van de code waarin de verschuivingseigenschappen zijn ingesteld (voor de regel `loader.filters = [displacementMap]`), ziet de afbeelding eruit alsof deze is uitgevlakt over het werkgebied:

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

De volgende code is een complexer voorbeeld waarbij een verschuivingskaartfilter wordt gebruikt om een vergrootglaseffect toe te passen op de afbeelding:

```
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);
```

De code genereert eerst twee verlopende cirkels die samen worden gecombineerd om de verschuivingskaartafbeelding te vormen. De rode cirkel veroorzaakt de verschuiving op de x-as (`xyFilter.componentX = BitmapDataChannel.RED`) en de blauwe cirkel veroorzaakt de verschuiving op de y-as (`xyFilter.componentY = BitmapDataChannel.BLUE`). De code voegt zowel de oorspronkelijke cirkels als de gecombineerde cirkel die fungeert als de kaartafbeelding onder in het scherm, toe om u te helpen begrijpen hoe de verschuivingskaartafbeelding eruit ziet.

De code laadt vervolgens een afbeelding en past, zodra de muis beweegt, het verschuivingsfilter toe op het deel van de afbeelding dat zich onder de muis bevindt. Door de verlopende cirkels die zijn gebruikt als de verschuivingskaartafbeelding, wordt het verschoven gebied van de muisaanwijzer vandaan verspreid. De grijze gebieden van de verschuivingskaartafbeelding veroorzaken geen verschuiving. De grijze kleur is `0x7F7F7F`. De blauwe en rode kanalen van deze grijstint komen precies overeen met de middelste tint van deze kleurkanalen, waardoor er geen verschuiving plaatsvindt in een grijs gebied van de kaartafbeelding. Op dezelfde manier vindt er geen verschuiving plaats in het midden van de cirkel. Hoewel de kleur daar niet grijs is, zijn de blauwe en rode kleurkanalen van die kleur gelijk aan de blauwe en rode kleurkanalen van middengrijs. Aangezien blauw en rood de kleuren zijn die de verschuiving veroorzaken, vindt er geen verschuiving plaats.

Arceringsfilter

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de klasse ShaderFilter kunt u een aangepast filtereffect gebruiken dat als een Pixel Bender-arcering is gedefinieerd. Het filtereffect wordt als Pixel Bender-arcering geschreven en kan daardoor volledig worden aangepast. De gefilterde inhoud wordt aan de arcering doorgegeven als afbeeldingsinvoer en het resultaat van de arceringsbewerking wordt het filterresultaat.

Opmerking: *Het arceerfilter is in ActionScript beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5.*

Als u een arceringsfilter op een object wilt toepassen, maakt u eerst een instantie Shader die de Pixel Bender-arcering vertegenwoordigt die u gebruikt. Zie "Werken met Pixel Bender-arceringen" op pagina 318 voor meer informatie over het maken van een Shader-instantie en hoe u invoerafbeeldingen en parameterwaarden opgeeft.

Wanneer u een arcering als filter gebruikt, moet u rekening houden met drie belangrijke zaken:

- De arcering moet gedefinieerd zijn voor het accepteren van ten minste één invoerafbeelding.

- Het gefilterde object (het weergaveobject of het object BitmapData waarop het filter wordt toegepast) wordt doorgegeven aan de arcering als eerste invoerafbeeldingswaarde. Hierdoor mag u niet handmatig een waarde opgeven voor de eerste afbeeldingsinvoer.
- Als de arcering meer dan één invoerafbeelding definieert, moeten de aanvullende invoerafbeeldingen handmatig worden opgegeven (dat gebeurt door middel van het instellen van de eigenschap `input` van een instantie ShaderInput die tot de instantie Shader behoort).

Wanneer u een object Shader voor de arcering hebt, kunt u een instantie ShaderFilter maken. Dit is het werkelijke filterobject dat u net als elk andere filter kunt gebruiken. Als u een ShaderFilter wilt maken dat een object Shader gebruikt, roept u de constructor `ShaderFilter()` aan en geeft u het object Shader door als argument, zoals hieronder aangegeven:

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

Zie "Arceringen als filter gebruiken" op pagina 336 voor een volledig voorbeeld van het gebruik van een arceringsfilter.

Voorbeeld van het filteren van weergaveobjecten: Filter Workbench

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De Filter Workbench biedt een gebruikersinterface voor het toepassen van verschillende filters op afbeeldingen en overige zichtbare inhoud en voor het weergeven van de resulterende code die kan worden gebruikt om hetzelfde effect te genereren in ActionScript. Naast gereedschap voor het experimenteren met filters, biedt deze toepassing de volgende technieken:

- Instanties maken van diverse filters
- Meerdere filters toepassen op een weergaveobject

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de toepassingsbestanden van Filter Workbench in de map Samples/FilterWorkbench. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	Klasse die de hoofdfunctionaliteit van de toepassing biedt, inclusief schakelen tussen inhoud waarop filters worden toegepast en het toepassen van filters op inhoud.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	Interface die de algemene methoden definieert die worden geïmplementeerd door elke filterfabrieksklassen. Deze interface definieert de algemene functionaliteit die de klasse FilterWorkbenchController gebruikt om te communiceren met de afzonderlijke filterfabrieksklassen.
In map com/example/programmingas3/filterWorkbench/: BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	Set klassen, die elk de interface IFilterFactory implementeert. Elk van deze klassen biedt de functionaliteit voor het maken en instellen van waarden voor een enkel type filter. De filtereigenschapdeelvensters in de toepassing gebruiken deze fabrieksklassen om instanties van hun specifieke filters te maken, die de klasse FilterWorkbenchController ophaalt en op de afbeeldingsinhoud toepast.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	Interface waarin algemene methoden worden gedefinieerd die worden geïmplementeerd door klassen die de deelvensters van de gebruikersinterface definiëren die worden gebruikt om filterwaarden in de toepassing te bewerken.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	Hulpprogrammaklasse die een methode bevat om een numerieke kleurwaarde om te zetten in een hexadecimale tekenreeksindeling.
com/example/programmingas3/filterWorkbench/GradientColor.as	Klasse die als waardeobject fungeert; deze maakt één enkel object van de drie waarden (kleur, alfa en verhouding) die zijn gekoppeld aan elke kleur in GradientBevelFilter en GradientGlowFilter.
Gebruikersinterface (Flex)	
FilterWorkbench.mxml	Het hoofdbestand dat de gebruikersinterface van de toepassing definieert.
flexapp/FilterWorkbench.as	Klasse die de functionaliteit biedt voor de gebruikersinterface van de hoofdtoepassing; deze klasse wordt gebruikt als code-achtergrondklasse voor het MXML-toepassingsbestand.

Bestand	Beschrijving
In map flexapp/filterPanels: BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	Set MXML-componenten die de functionaliteit biedt voor elk deelvenster dat wordt gebruikt om opties in te stellen voor een enkel filter.
flexapp/ImageContainer.as	Een weergaveobject dat fungeert als container voor de geladen afbeelding op het scherm.
flexapp/controls/BGColorCellRenderer.as	Aangepaste celrenderer die wordt gebruikt om de achtergrondkleur van een cel in de component DataGrid te wijzigen.
flexapp/controls/QualityComboBox.as	Aangepast besturingselement dat een keuzelijst definieert die kan worden gebruikt voor de instelling Quality in verschillende filterdeelvensters.
flexapp/controls/TypeComboBox.as	Aangepast besturingselement dat een keuzelijst definieert die kan worden gebruikt voor de instelling Type in verschillende filterdeelvensters.
Gebruikersinterface (Flash)	
FilterWorkbench.fla	Het hoofdbestand dat de gebruikersinterface van de toepassing definieert.
flashapp/FilterWorkbench.as	Klasse die de functionaliteit biedt voor de gebruikersinterface van de hoofdtoepassing; deze klasse wordt gebruikt als documentklasse voor het FLA-toepassingsbestand.
In map flashapp/filterPanels: BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	Set klassen die de functionaliteit biedt voor elk deelvenster dat wordt gebruikt om opties in te stellen voor een enkel filter. Voor elke klasse bestaat er ook een gekoppeld filmclipsymbool in de bibliotheek van het FLA-hoofdtoepassingsbestand, waarvan de naam overeenkomt met de naam van de klasse (het symbool BlurPanel is bijvoorbeeld gekoppeld aan de klasse die is gedefinieerd in BlurPanel.as). De componenten die de gebruikersinterface vormen, worden gepositioneerd en benoemd binnen deze symbolen.
flashapp/ImageContainer.as	Een weergaveobject dat fungeert als container voor de geladen afbeelding op het scherm.
flashapp/BGColorCellRenderer.as	Aangepaste celrenderer die wordt gebruikt om de achtergrondkleur van een cel in de component DataGrid te wijzigen.

Bestand	Beschrijving
flashapp/ButtonCellRenderer.as	Aangepaste celrenderer die wordt gebruikt om een component Button op te nemen in een cel in de component DataGrid.
Gefilterde afbeeldingsinhoud	
com/example/programmingas3/filterWorkbench/ImageType.as	Deze klasse fungeert als een waardeobject dat het type en de URL bevat van een enkel afbeeldingsbestand waarnaar de toepassing filters kan laden en toepassen. De klasse bevat tevens een set constanten die de werkelijk beschikbare afbeeldingsbestanden vertegenwoordigt.
images/sampleAnimation.swf images/sampleImage1.jpg images/sampleImage2.jpg	Afbeeldingen en overige zichtbare inhoud waarop filters worden toegepast in de toepassing.

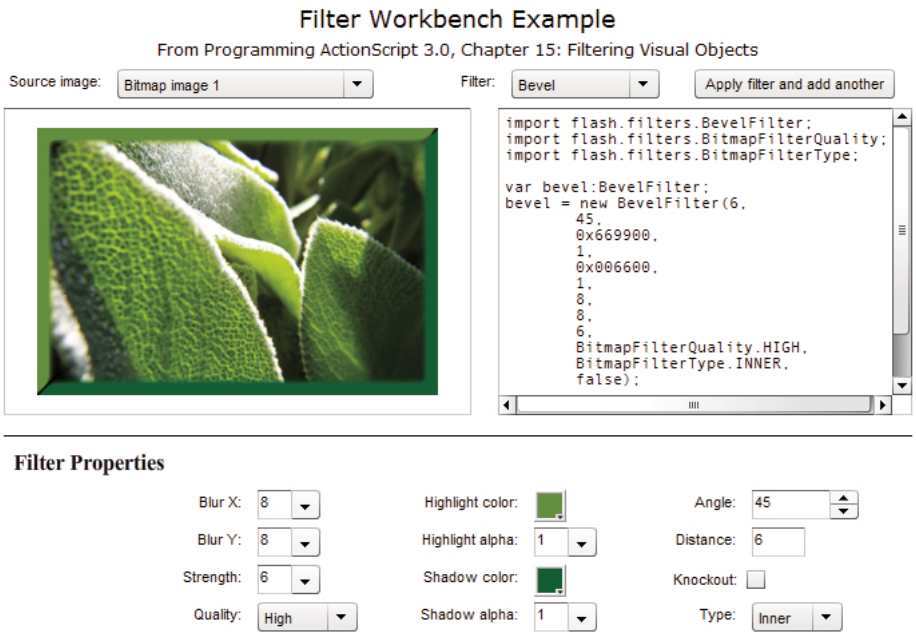
Experimenteren met ActionScript-filters

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing Filter Workbench is ontworpen om u te helpen experimenteren met diverse filtereffecten en het genereren van de relevante ActionScript-code voor die effecten. U kunt kiezen uit drie verschillende bestanden met zichtbare inhoud (inclusief bitmapafbeeldingen en een animatie die met Flash is gemaakt) en u kunt acht verschillende ActionScript-filters toepassen op de geselecteerde afbeelding, zowel afzonderlijk als in combinatie met andere filters. De toepassing bevat de volgende filters:

- Schuine-kantfilter (flash.filters.BevelFilter)
- Vervagend filter (flash.filters.BlurFilter)
- Kleurenmatrixfilter (flash.filters.ColorMatrixFilter)
- Convolutiefilter (flash.filters.ConvolutionFilter)
- Slagschaduwfilter (flash.filters.DropShadowFilter)
- Gloedfilter (flash.filters.GlowFilter)
- Verlopende schuine-kantfilter (flash.filters.GradientBevelFilter)
- Verlopende-gloedfilter (flash.filters.GradientGlowFilter)

Zodra een gebruiker een afbeelding heeft geselecteerd en een filter op die afbeelding heeft toegepast, wordt een deelvenster met besturingselementen weergegeven voor het instellen van de specifieke eigenschappen van het geselecteerde filter. De volgende afbeelding toont bijvoorbeeld de toepassing met een geselecteerd schuine-kantfilter:



Terwijl de gebruiker de filtereigenschappen aanpast, wordt de voorvertoning realtime bijgewerkt. De gebruiker kan ook meerdere filters toepassen door een filter aan te passen, te klikken op de knop Apply filter and add another, een volgend filter aan te passen, nogmaals op de knop Apply filter and add another te klikken enzovoort.

De filterdeelvensters van de toepassing bevatten een aantal functies en beperkingen:

- Het kleurenmatrixfilter bevat een set besturingselementen voor het rechtstreeks manipuleren van algemene afbeeldingseigenschappen, zoals helderheid, contrast, verzadiging en kleurtoon. Bovendien kunnen aangepaste kleurenmatrixwaarden worden opgegeven.
- Het convolutiefilter, dat alleen beschikbaar is in ActionScript, bevat een set veelgebruikte convolutiematrixwaarden, maar er kunnen ook aangepaste waarden worden opgegeven. De klasse ConvolutionFilter accepteert een matrix met willekeurige grootte, maar de toepassing Filter Workbench gebruikt standaard een matrix van 3 x 3, de meestgebruikte filtergrootte.
- Het verschuivingskaartfilter en het arceringsfilter, alleen beschikbaar in ActionScript, zijn niet beschikbaar in de toepassing Filter Workbench.

Filterinstanties maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing Filter Workbench bevat een set klassen, één voor elk van de beschikbare filters, die worden gebruikt door de afzonderlijke deelvensters om de filters te maken. Wanneer een gebruiker een filter selecteert, maakt de ActionScript-code die is gekoppeld aan het filterdeelvenster een instantie van de juiste filterfabrieksklasse. (Deze klassen staan bekend als *fabrieksklassen* omdat zij instanties van andere objecten maken, zoals een fabriek afzonderlijke producten maakt.)

Op het moment dat de gebruiker een eigenschapwaarde in het deelvenster wijzigt, wordt aan de hand van de code van het deelvenster de juiste methode in de fabrieksklasse aangeroepen. Elke fabrieksklasse bevat specifieke methoden die het deelvenster gebruikt om de juiste filterinstantie te maken. Wanneer de gebruiker bijvoorbeeld het vervagende filter selecteert, maakt de toepassing een instantie van BlurFactory. De klasse BlurFactory bevat een methode `modifyFilter()` die drie parameters accepteert: `blurX`, `blurY` en `quality`, die tezamen worden gebruikt om de gewenste instantie BlurFilter te maken:

```
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));
}
```

Wanneer de gebruiker daarentegen het convolutiefilter selecteert, heeft de gebruiker meer flexibiliteit ter beschikking en dus een grotere set eigenschappen om te beheren. In de klasse ConvolutionFactory wordt de volgende code aangeroepen wanneer de gebruiker een andere waarde in het filterdeelvenster selecteert:

```
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));
}
```

Telkens wanneer de filterwaarden worden gewijzigd, verzendt het fabrieksobject een gebeurtenis `Event.CHANGE` om aan listeners te melden dat de waarden van het filter zijn gewijzigd. De klasse FilterWorkbenchController, die de filters werkelijk op de gefilterde inhoud toepast, luistert naar die gebeurtenis om te bepalen wanneer deze een nieuwe kopie van het filter moet ophalen en opnieuw op de gefilterde inhoud moet toepassen.

De klasse FilterWorkbenchController hoeft geen specifieke details van elke filterfabrieksklasse te weten, maar moet wel weten dat het filter is gewijzigd en moet een kopie van het filter kunnen benaderen. Hiertoe bevat de toepassing een interface, IFilterFactory, die het gedrag definieert dat een filterfabrieksklasse moet bieden zodat de instantie FilterWorkbenchController van de toepassing correct kan functioneren. IFilterFactory definieert de methode `getFilter()` die wordt gebruikt in de klasse FilterWorkbenchController:

```
function getFilter():BitmapFilter;
```

De interfacemethodedefinitie `getFilter()` geeft op dat een instantie BitmapFilter moet worden geretourneerd in plaats van een specifiek type filter. De klasse BitmapFilter definieert geen specifiek type filter. In plaats daarvan is BitmapFilter de basisklasse voor alle filterklassen. Elke filterfabrieksklasse definieert een specifieke implementatie van de methode `getFilter()` waarin een verwijzing wordt geretourneerd naar het filterobject dat is gebouwd. Hier ziet u bijvoorbeeld een verkorte versie van de broncode van de klasse ConvolutionFactory:

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

In de implementatie van de klasse ConvolutionFactory van de methode `getFilter()` wordt een instantie ConvolutionFilter geretourneerd, hoewel elk object dat `getFilter()` aanroept niet noodzakelijkerwijs hoeft te weten dat (volgens de definitie van de methode `getFilter()` waarnaar ConvolutionFactory handelt) een instantie BitmapFilter moet worden geretourneerd, waarbij het niet uitmaakt van welke ActionScript-filterklasse deze afkomstig is.

Filters toepassen op weergaveobjecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals is beschreven in de vorige sectie, gebruikt de toepassing Filter Workbench een instantie van de klasse FilterWorkbenchController (waarnaar verder wordt verwezen als de 'bedieningsinstantie'), die de filters daadwerkelijk op het geselecteerde zichtbare object toepast. Voordat de bedieningsinstantie een filter kan toepassen, moet deze eerst weten op welke afbeelding of zichtbare inhoud het filter moet worden toegepast. Wanneer de gebruiker een afbeelding selecteert, roept de toepassing de methode `setFilterTarget()` in de klasse FilterWorkbenchController aan, waarbij één van de constanten wordt doorgegeven die in de klasse ImageType is gedefinieerd:

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

Met behulp van die informatie laadt de bedieningsinstantie het opgegeven bestand en slaat dit op in een instantievariabele met de naam `_currentTarget` zodra het is geladen:

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

Wanneer de gebruiker een filter selecteert, roept de toepassing de methode `setFilter()` van de bedieningsinstantie aan en geeft hierbij een verwijzing aan de bedieningsinstantie door naar het relevante filterfabrieksobject, die het opslaat in een instantievariabele met de naam `_filterFactory`.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

Zoals eerder beschreven, weet de bedieningsinstantie niet wat het specifieke gegevenstype is van de filterfabrieksinstantie die eraan wordt gegeven; de instantie weet alleen dat het object de instantie IFilterFactory implementeert, met andere woorden, het heeft een methode `getFilter()` en verzendt een gebeurtenis `change` (`Event.CHANGE`) wanneer het filter wordt gewijzigd.

Wanneer de gebruiker de eigenschappen van een filter wijzigt in het filterdeelvenster, ziet de bedieningsinstantie dat het filter is gewijzigd via de filterfabrieksgebeurtenis `change`, die de methode `filterChange()` van de bedieningsinstantie aanroept. Die methode roept vervolgens de methode `applyTemporaryFilter()` aan:

```
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();
}
```

Het toepassen van het filter op het weergaveobject gebeurt binnen de methode `applyTemporaryFilter()`. De bedieningsinstantie haalt eerst een verwijzing op naar het filterobject door de filterfabrieksmethode `getFilter()` aan te roepen.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

De bedieningsinstantie bevat een instantievariabele Array met de naam `_currentFilters`, waarin deze alle filters opslaat die op het weergaveobject zijn toegepast. De volgende stap bestaat uit het toevoegen van het recent bijgewerkte filter aan die array:

```
_currentFilters.push(currentFilter);
```

Vervolgens wijst de code de array van `filters` toe aan de eigenschap filters van het weergaveobject, die de filters daadwerkelijk op de afbeelding toepast:

```
_currentTarget.filters = _currentFilters;
```

Ten slotte, omdat het laatst toegevoegde filter nog steeds het 'werkende' filter is, moet het niet permanent worden toegepast op het weergaveobject, daarom wordt het verwijderd uit de array `_currentFilters`:

```
_currentFilters.pop();
```

Het verwijderen van dit filter uit de array heeft geen effect op het gefilterde weergaveobject, omdat een weergaveobject een kopie maakt van de array filters wanneer deze wordt toegewezen aan de eigenschap `filters` en het gebruikt die interne array in plaats van de oorspronkelijke array. Daarom hebben wijzigingen in de array van filters geen effect op het weergaveobject totdat de array nogmaals wordt toegewezen aan de eigenschap `filters` van het weergaveobject.

Hoofdstuk 15: Werken met Pixel Bender-arceringen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de Adobe Pixel Bender-toolkit kunnen ontwikkelaars arceringen schrijven waarmee grafische effecten worden bereikt en waarmee ook andere bewerkingen op afbeeldingen en gegevens kunnen worden uitgevoerd. De Pixel Bender-bytecode kan in ActionScript worden uitgevoerd om het effect op afbeeldingsgegevens of visuele inhoud toe te passen. Met Pixel Bender-arceringen in ActionScript kunt u aangepaste visuele effecten maken en gegevens verwerken die verder gaan dan de ingebouwde mogelijkheden in ActionScript.

Opmerking: *In Flash Player 10 of hoger en in Adobe AIR 1.5 of hoger wordt ondersteuning geboden voor Pixel Bender. Pixel Bender-overvloeiingen, -filters en -vullingen worden niet ondersteund in combinatie met GPU-rendering.*

Meer Help-onderwerpen

Adobe Pixel Bender Technology Center

Pixel Bender Developer's Guide (Ontwikkelaargids voor Pixel Bender)

Pixel Bender Reference

flash.display.Shader

flash.filters.ShaderFilter

Basisbeginselen van Pixel Bender voor Flash

Basisbeginselen van Pixel Bender voor Flex

Basisbeginselen van Pixel Bender-arceringen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Adobe Pixel Bender is een programmeertaal voor het maken en bewerken van grafische inhoud. Met Pixel Bender maakt u een kernel, die in dit document ook arcering wordt genoemd. Met de arcering definieert u één functie die op elke pixel van een afbeelding afzonderlijk wordt uitgevoerd. Het resultaat van elke oproep van de functie is de uitvoerkleur bij de desbetreffende pixelcoördinaat in de afbeelding. U kunt invoerafbeeldingen en parameterwaarden opgeven om de bewerking aan te passen. In één uitvoering van een arcering zijn de invoer- en parameterwaarden constant. Het enige wat varieert, is de coördinaat van de pixel waarvan de kleur het resultaat van de functieoproep is.

Waar mogelijk wordt de arceringsfunctie voor meerdere pixelcoördinaten parallel opgeroepen. Hierdoor verbeteren de prestaties van de arcering en kunnen bewerkingen zeer snel worden uitgevoerd.

In ActionScript kunnen drie soorten effecten eenvoudig worden gemaakt met een arcering:

- vulling bij het tekenen
- overvloeimodus
- filter

Een arcering kan ook in de zelfstandige modus worden uitgevoerd. In de zelfstandige modus wordt het resultaat van een arcering direct benaderd en wordt niet het bedoelde gebruik van de arcering vooraf opgegeven. Het resultaat kan worden benaderd als grafische gegevens of als binaire of numerieke gegevens. De gegevens hoeven geen grafische gegevens te zijn. Zo kunt u aan een arcering een set gegevens als invoer geven. De arcering bewerkt de gegevens en u kunt de resulterende gegevens benaderen die worden geretourneerd door de arcering.

Pixel Bender-ondersteuning is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5. Pixel Bender-overvloeiingen, -filters en -vullingen worden niet ondersteund onder GPU-rendering. Op mobiele apparaten worden Pixel Bender-arceringen wel uitgevoerd onder CPU-rendering. De prestaties vallen echter tegen in vergelijking met die van desktopcomputers. Vele arceringsprogamma's kunnen slechts een paar frames per seconde uitvoeren.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het maken en gebruiken van Pixel Bender-arceringen:

Kernel in Pixel Bender is een kernel hetzelfde als een arcering. Met Pixel Bender legt u met uw code een kernel vast waarin één functie is gedefinieerd die op elke pixel van een afbeelding afzonderlijk wordt uitgevoerd.

Pixel Bender-bytecode Wanneer een Pixel Bender-kernel wordt gecompileerd, wordt deze omgezet in Pixel Bender-bytecode. De bytecode wordt tijdens runtime benaderd en uitgevoerd.

Pixel Bender-taal De programmeertaal die wordt gebruikt om een Pixel Bender-kernel te maken.

Pixel Bender-toolkit De toepassing waarmee een bestand met Pixel Bender-bytecode wordt gemaakt van Pixel Bender-broncode. Met de toolkit kunt u Pixel Bender-broncode schrijven, testen en compileren.

Arcering In de context van dit document is een arcering een functieset die is geschreven in de Pixel Bender-taal. De code van een arcering maakt een visueel effect of voert een berekening uit. In beide gevallen retourneert de arcering een set gegevens (doorgaans de pixels van een afbeelding). De arcering voert voor elk gegevenspunt dezelfde bewerking uit, met als enige verschil de coördinaten van de uitvoerpixel. De arcering wordt niet in ActionScript geschreven. Een arcering wordt geschreven in de Pixel Bender-taal en gecompileerd naar Pixel Bender-bytecode. Een arcering kan bij de compilatie worden ingesloten in een SWF-bestand of tijdens de runtime worden geladen als extern bestand. In beide gevallen wordt de arcering in ActionScript benaderd door een Shader-object te maken en dit object te koppelen aan de bytecode van de arcering.

Arceringsinvoer Complexe invoer, doorgaans grafische bitmapgegevens, die aan een arcering wordt doorgegeven om te gebruiken in de berekeningen. Voor elke invoervariabele die in een arcering is gedefinieerd, wordt één waarde (dat wil zeggen één afbeelding of set binaire gegevens) gebruikt voor de hele uitvoering van de arcering.

Arceringsparameter Eén waarde (of een beperkte set waarden) die aan een arcering wordt doorgegeven om te gebruiken in de berekeningen. Elke parameterwaarde wordt gedefinieerd voor één uitvoering van de arcering, en dezelfde waarde wordt tijdens de hele uitvoering van de arcering gebruikt.

Codevoorbeelden binnen dit hoofdstuk doorlopen

Wanneer u het hoofdstuk doorwerkt, wilt u de codevoorbeelden wellicht uittesten. Bij het testen moet de code worden uitgevoerd en moeten de resultaten worden bekeken in het nieuwe SWF-bestand. In alle voorbeelden wordt inhoud gemaakt met de API voor tekenen die wordt gebruikt of veranderd door het arceringseffect.

De meeste codevoorbeelden bestaan uit twee delen. Eén deel is de Pixel Bender-broncode voor de arcering die wordt gebruikt in het voorbeeld. U moet eerst met de Pixel Bender Toolkit de broncode compileren naar een bestand met Pixel Bender-bytecode. Voer de volgende stappen uit om het bestand met Pixel Bender-bytecode te maken:

1. Open Adobe Pixel Bender Toolkit. Kies zo nodig in het menu Build de opdracht Turn on Flash Player warnings and errors.

2 Kopieer de Pixel Bender-code en plak deze in het deelvenster met de code-editor van de Pixel Bender Toolkit.

3 Kies in het menu File de opdracht Export kernel filter for Flash Player.

4 Sla het bestand met Pixel Bender-bytecode op in dezelfde map als het Flash-document. De bestandsnaam moet gelijk zijn aan de naam in het voorbeeld.

Het ActionScript-deel van elk voorbeeld wordt geschreven als klassebestand. Ga als volgt te werk om het voorbeeld te testen in Flash Professional:

1 Maak een leeg Flash-document en sla dit op uw computer op.

2 Maak een nieuw ActionScript-bestand en sla het op in dezelfde map als het Flash-document. De bestandsnaam moet gelijk zijn aan de naam van de klasse in het codevoorbeeld. Als in de code bijvoorbeeld een klasse met de naam MyApplication wordt gedefinieerd, slaat u het ActionScript-bestand op als MyApplication.as.

3 Kopieer de code naar het ActionScript-bestand en sla het bestand op.

4 Klik in het Flash-document op een leeg deel van het werkgebied of de werkruimte om Eigenschapcontrole van het document te activeren.

5 Typ in het veld Documentklasse in Eigenschapcontrole de naam van de ActionScript-klasse die u uit de tekst hebt overgenomen.

6 Voer het programma uit via Besturing > Film testen

De resultaten van het voorbeeld worden weergegeven in het voorbeeldvenster.

Een uitgebreide beschrijving van deze technieken voor het testen van codevoorbeelden is te vinden in "ActionScript-voorbeelden gebruiken" op pagina 1167.

Arceringen laden of insluiten

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als u werkt met een Pixel Bender-arcering in ActionScript moet u eerst toegang hebben tot de arcering in de ActionScript-code. Aangezien een arcering met de toolkit van Adobe Pixel Bender wordt gemaakt en is geschreven in de Pixel Bender-taal, hebt u vanuit ActionScript niet rechtstreeks toegang tot de arcering. In plaats hiervan maakt u een instantie van de Shader-klasse die de Pixel Bender-arcering in ActionScript vertegenwoordigt. Met het Shader-object kunt u meer informatie over de arcering krijgen, zoals over mogelijke parameters die worden verwacht of waarden van invoerafbeeldingen. U geeft het Shader-object door aan andere objecten om de arcering daadwerkelijk te gebruiken. Als u de arcering bijvoorbeeld als filter wilt gebruiken, moet u het Shader-object toewijzen aan de eigenschap `shader` van een ShaderFilter-object. Als u de arcering als vulling bij het tekenen wilt gebruiken, moet u het Shader-object als argument doorgeven aan de methode `Graphics.beginShaderFill()`.

Via de ActionScript-code kunt u op twee manieren toegang krijgen tot een arcering die met de toolkit van Adobe Pixel Bender (een PJB-bestand) is gemaakt:

- Geladen bij uitvoering: het arceringsbestand kan met een URLLoader-object worden geladen als een extern element. Deze techniek is vergelijkbaar met het laden van een extern element, zoals een tekstbestand. In het volgende voorbeeld wordt een bytecodebestand voor een arcering in runtime geladen en gekoppeld aan een Shader-instantie:

```
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
}
```

- Ingesloten in het SWF-bestand: het arceringsbestand kan tijdens het compileren in het SWF-bestand worden ingesloten met de metagegevenstag `[Embed]`. De metagegevenstag `[Embed]` is alleen beschikbaar als u het SWF-bestand compileert met de Flex SDK. Voor de tag `[Embed]` geldt dat de parameter `source` verwijst naar het arceringsbestand en dat de parameter `mimeType"application/octet-stream"` is, zoals in dit voorbeeld:

```
[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
```

In beide gevallen koppelt u de bytecode van de onbewerkte arcering (de eigenschap `URLLoader.data` of een instantie van de gegevensklasse `[Embed]`) aan de Shader-instantie. Zoals de vorige voorbeelden laten zien, kunt u de bytecode op twee manieren aan de Shader-instantie toewijzen. U kunt de bytecode van de arcering als een argument doorgeven aan de constructor `Shader()`. U kunt deze ook instellen als de eigenschap `byteCode` van de Shader-instantie.

Als u eenmaal een Pixel Bender-arcering hebt gemaakt en aan een Shader-object hebt gekoppeld, kunt u de arcering gebruiken om op verschillende manieren effecten te maken. U kunt de arcering als filter, overvloeimodus of bitmapvulling gebruiken of gebruiken voor het zelfstandig verwerken van bitmapgegevens of andere gegevens. U kunt de eigenschap `data` van het Shader-object ook gebruiken om toegang te krijgen tot de metagegevens van de arcering, om invoerafbeeldingen op te geven en om parameterwaarden in te stellen.

Toegang krijgen tot de metagegevens van arceringen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Tijdens het maken van een Pixel Bender-arceringskernel, kan de auteur metagegevens over de arcering opgeven in de broncode van Pixel Bender. Tijdens het gebruik van een arcering in ActionScript kunt u de arcering bekijken en de metagegevens van de arcering extraheren.

Wanneer u een Shader-instantie maakt en deze instantie aan een Pixel Bender-arcering koppelt, wordt er een ShaderData-object met gegevens over de arcering gemaakt en opgeslagen in de eigenschap `data` van het Shader-object. In de ShaderData-klasse zelf worden geen eigenschappen gedefinieerd. Tijdens de uitvoering wordt echter wel dynamisch een eigenschap aan het ShaderData-object toegevoegd voor elke metagegevenswaarde die in de broncode van de arcering is gedefinieerd. De naam die aan elke eigenschap wordt gegeven, is gelijk aan de naam die in de metagegevens is opgegeven. Stel dat in de broncode van een Pixel Bender-arcering de volgende metagegevensdefinitie is opgenomen:

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

Het ShaderData-object dat voor die arcering is gemaakt, wordt gemaakt met de volgende eigenschappen en waarden:

- `namespace` (String): `"Adobe::Example"`
- `vendor` (String): `"Bob Jones"`
- `version` (String): `"1"`
- `description` (String): `"Creates a version of the specified image with the specified brightness"`

Omdat eigenschappen van metagegevens dynamisch aan het ShaderData-object worden toegevoegd, kunt u een `for..in`-lus gebruiken om het ShaderData-object te bekijken. Via deze techniek kunt u te weten komen of de arcering over metagegevens beschikt en wat de waarden van de metagegevens zijn. Naast de eigenschappen van de metagegevens kan een ShaderData-object ook eigenschappen bevatten die invoerwaarden en parameters vertegenwoordigen die in de arcering zijn gedefinieerd. Wanneer u een `for..in`-lus gebruikt om een ShaderData-object te bekijken, moet u het gegevenstype van elke eigenschap controleren om vast te stellen of het bij de eigenschap om invoer (een ShaderInput-instantie), een parameter (een ShaderParameter-instantie) of een metagegevenswaarde (een String-instantie) gaat. In het volgende voorbeeld wordt getoond hoe u een `for..in`-lus gebruikt om de dynamische eigenschappen van de eigenschap`data` van een arcering te bekijken. Elke metagegevenswaarde wordt toegevoegd aan een Vector-instantie met de naam `metadata`. In dit voorbeeld wordt ervan uitgegaan dat er al een Shader-instantie met de naam `myShader` is gemaakt:

```
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
```

Zie "Invoer en parameters van arceringen identificeren" op pagina 323 voor een versie van dit voorbeeld waarmee ook invoerwaarden en parameters van arceringen worden geëxtraheerd. Zie "Waarden voor invoer en parameters van arceringen opgeven" op pagina 323 voor meer informatie over eigenschappen voor invoer en parameters.

Waarden voor invoer en parameters van arceringen opgeven

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Veel Pixel Bender-arceringen worden zodanig gedefinieerd dat er een of meer invoerafbeeldingen worden gebruikt bij het verwerken van de arcering. Het is bijvoorbeeld heel gewoon voor een arcering om een bronafbeelding te accepteren en die afbeelding weer te geven met een bepaald effect. De manier waarop de arcering wordt gebruikt bepaalt of de invoerwaarde automatisch kan worden opgegeven of dat u zelf een waarde moet opgeven. Door veel arceringen worden op een vergelijkbare manier parameters opgegeven die worden gebruikt om de uitvoer van de arcering aan te passen. U moet ook expliciet een waarde instellen voor elke parameter voordat u de arcering gebruikt.

U gebruikt de eigenschap `data` van het Shader-object om invoer en parameters voor de arcering in te stellen en om na te gaan of een bepaalde arcering invoer of parameters verwacht. De eigenschap `data` is een ShaderData-instantie.

Invoer en parameters van arceringen identificeren

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Bij het identificeren van invoer en parameters van arceringen moet u eerst nagaan of de specifieke arcering die u gebruikt, invoerafbeeldingen of parameters verwacht. Elke Shader-instantie heeft een `data`-eigenschap met een ShaderData-object. Als in de arcering invoer of parameters worden gedefinieerd, worden ze gebruikt als eigenschappen van dat ShaderData-object. De namen van de eigenschappen komen overeen met de namen die voor de invoer en de parameters in de broncode van de arcering zijn opgegeven. Als in een arcering bijvoorbeeld een invoer met de naam `src` wordt gedefinieerd, bevat het ShaderData-object een eigenschap met de naam `src` die staat voor de desbetreffende invoer. Elke eigenschap die een invoer vertegenwoordigt, is een ShaderInput-instantie en elke eigenschap die een parameter vertegenwoordigt is een ShaderParameter-instantie.

In het gunstigste geval biedt de auteur van de arcering documentatie voor de arcering, waarin de waarden voor invoerafbeeldingen en parameters worden aangegeven die door de de arcering worden verwacht, waar deze waarden voor staan, wat de bijbehorende waarden zijn, enzovoort.

Als de arcering echter niet wordt gedocumenteerd (en u beschikt niet over de broncode van de arcering), kunt u de arceringsgegevens inspecteren om de invoerwaarden en parameters te identificeren. De eigenschappen die de invoerwaarden en parameters vertegenwoordigen, worden dynamisch toegevoegd aan het ShaderData-object. U kunt daarom een `for..in`-lus gebruiken om het ShaderData-object te bekijken om na te gaan of in de aan het object gekoppelde arcering invoerwaarden of parameters zijn gedefinieerd. Zoals werd beschreven in "Toegang krijgen tot de metagegevens van arceringen" op pagina 322 wordt elke metagegevenswaarde die voor een arcering is gedefinieerd, ook opgehaald als een dynamische eigenschap die is toegevoegd aan de eigenschap`Shader.data`. Wanneer u deze techniek gebruikt om invoer en parameters van arceringen te identificeren, moet u het gegevenstype van de dynamische eigenschappen controleren. Als het bij een eigenschap om een ShaderInput-instantie gaat, vertegenwoordigt de eigenschap een invoerwaarde. Als het een ShaderParameter-instantie is, vertegenwoordigt de eigenschap een parameter. In alle andere gevallen is de eigenschap een metagegevenswaarde. In het volgende

voorbeeld wordt getoond hoe u een `for..in`-lus gebruikt om de dynamische eigenschappen van de eigenschap`data` van een arcering te bekijken. Elke invoerwaarde (ShaderInput-object) wordt toegevoegd aan een Vector-instantie met de naam `inputs`. Elke parameter (ShaderParameter-object) wordt toegevoegd aan een Vector-instantie met de naam `parameters`. Eventuele eigenschappen voor metagegevens worden toegevoegd aan een Vector-instantie met de naam `metadata`. In dit voorbeeld wordt ervan uitgegaan dat er al een Shader-instantie met de naam `myShader` is gemaakt:

```
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
```

Invoerwaarden voor arceringen opgeven

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Veel arceringen verwachten een of meer invoerafbeeldingen die worden gebruikt bij het verwerken van de arcering. In veel gevallen wordt een invoerwaarde echter automatisch opgegeven wanneer het Shader-object wordt gebruikt. Stel dat een arcering één invoerwaarde nodig heeft en dat de arcering als filter wordt gebruikt. Wanneer het filter op een weergaveobject of BitmapData-object wordt toegepast, wordt dat object automatisch als invoerwaarde ingesteld. In dat geval stelt u niet expliciet een invoerwaarde in.

In sommige gevallen moet u echter wel specifiek een invoerwaarde instellen, met name als in een arcering meerdere invoerwaarden zijn gedefinieerd. Elke invoerwaarde die in een arcering is gedefinieerd, wordt in ActionScript vertegenwoordigd door een ShaderInput-object. Het ShaderInput-object is een eigenschap van de ShaderData-instantie in de eigenschap `data` van het Shader-object, zoals wordt beschreven in "Invoer en parameters van arceringen identificeren" op pagina 323. Stel dat in een arcering een invoerwaarde met de naam `src` wordt gedefinieerd en dat die arcering is gekoppeld aan een Shader-object met de naam `myShader`. In dat geval gebruikt u de volgende id om toegang te krijgen tot het ShaderInput-object dat overeenkomt met de invoerwaarde `src`:

```
myShader.data.src
```

Elk ShaderInput-object heeft een eigenschap `input` waarmee de waarde voor de invoer wordt ingesteld. U stelt de eigenschap `input` in op een BitmapData-instantie om afbeeldingsgegevens op te geven. U kunt de eigenschap `input` ook instellen op een BitmapData of Vector.<Number>-instantie om binaire gegevens of getalsgegevens op te geven. Zie voor meer gegevens en beperkingen over het gebruik van een BitmapData of Vector.<Number>-instantie als invoergegeven de `ShaderInput.input`-vermelding in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform.

Naast de eigenschap `input` heeft een ShaderInput-object eigenschappen waarmee het type afbeelding kan worden bepaald dat door de invoer wordt verwacht. Hierbij gaat het onder andere om de eigenschappen `width`, `height` en `channels`. Elk ShaderInput-object heeft ook een eigenschap `index`, een handige eigenschap om te bepalen of een expliciete waarde voor de invoer moet worden gegeven. Als een arcering meer invoerwaarden verwacht dan het automatisch ingestelde aantal, kunt u de waarden van die invoer instellen. Zie "Een arcering gebruiken" op pagina 328 voor informatie over de verschillende manieren waarop u arceringen kunt gebruiken en om na te gaan of invoerwaarden automatisch worden ingesteld.

Parameterwaarden voor arceringen opgeven

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Bij sommige arceringen worden parameterwaarden gedefinieerd waarmee het gewenste resultaat wordt bereikt. Zo kunt u bij een arcering waarmee de helderheid van een afbeelding wordt gewijzigd, bijvoorbeeld een parameter voor helderheid opgeven die aangeeft in hoeverre de helderheid wordt beïnvloed door de bewerking. Een enkele parameter die in een arcering wordt gedefinieerd, kan een enkele of meerdere waarden verwachten, afhankelijk van de parameterdefinitie in de arcering. Elke parameter die in een arcering is gedefinieerd, wordt in ActionScript vertegenwoordigd door een ShaderParameter-object. Het ShaderParameter-object is een eigenschap van de ShaderData-instantie in de eigenschap data van het Shader-object, zoals wordt beschreven in "Invoer en parameters van arceringen identificeren" op pagina 323. Stel dat in een arcering bijvoorbeeld de parameter `brightness` wordt gedefinieerd en dat die arcering wordt vertegenwoordigd door een Shader-object met de naam `myShader`. In dat geval gebruikt u de volgende id om toegang te krijgen tot het ShaderParameter-object dat overeenkomt met de parameter `brightness`:

```
myShader.data.brightness
```

Als u een waarde (of waarden) wilt opgeven voor de parameter, maakt u een ActionScript-array die de waarde of waarden bevat en wijst u deze array toe aan de eigenschap `value` van het ShaderParameter-object. De eigenschap `value` is gedefinieerd als een Array-instantie omdat één arceringsparameter mogelijk meerdere waarden vereist. Zelfs als de arceringsparameter slechts één waarde verwacht, moet u de waarde in een Array-object opnemen om de waarde aan de eigenschap `ShaderParameter.value` toe te kunnen wijzen. In de volgende code wordt getoond hoe een enkele waarde wordt ingesteld als de eigenschap `value`:

```
myShader.data.brightness.value = [75];
```

Als in de broncode van de Pixel Bender voor de arcering een standaardwaarde voor de parameter is gedefinieerd, wordt er een array met de standaardwaarde of -waarden gemaakt en toegewezen aan de eigenschap `value` van het ShaderParameter-object wanneer het Shader-object wordt gemaakt. Als er eenmaal een array aan de eigenschap `value` is toegewezen (ook als het de standaardarray is), kan de parameterwaarde worden gewijzigd door de waarde van het array-element te wijzigen. U hoeft geen nieuwe array te maken en deze toe te wijzen aan de eigenschap `value`.

In het volgende voorbeeld ziet u hoe u de parameterwaarde van een arcering in ActionScript instelt. In dit voorbeeld wordt een parameter met de naam `color` gedefinieerd. De parameter `color` wordt gedeclareerd als een `float4`-variabele in de broncode van Pixel Bender, wat inhoudt dat het een array van vier getallen met drijvende komma is. In het voorbeeld wordt de waarde van de parameter `color` voortdurend gewijzigd en telkens wanneer de parameterwaarde wordt gewijzigd, wordt de arcering gebruikt om een gekleurde rechthoek op het scherm te tekenen. Het resultaat is een bewegende kleurverandering.

Opmerking: *De code voor dit voorbeeld is geschreven door Ryan Taylor. Onze dank gaat uit naar Ryan, die dit voorbeeld met ons deelt. U kunt de portfolio van Ryan bekijken en zijn codes lezen op www.boostworthy.com/.*

Het draait bij de ActionScript-code om drie methoden:

- `init()`: in de methode `init()` laadt de code het bestand met Pixel Bender-bytecode dat de arcering bevat. Wanneer het bestand is geladen, wordt de methode `onLoadComplete()` aangeroepen.
- `onLoadComplete()`: in de methode `onLoadComplete()` maakt de code het Shader-object `shader`. De code maakt ook de nieuwe Sprite-instantie `texture`. In de methode `renderShader()` tekent de code het resultaat van de arcering één keer per frame in `texture`.
- `onEnterFrame()`: De methode `onEnterFrame()` wordt één keer per frame aangeroepen, waarbij het animatie-effect wordt gemaakt. In deze methode wordt de waarde van de arceringsparameter op de nieuwe kleur ingesteld, waarna de methode `renderShader()` wordt aangeroepen om het arceringsresultaat als een rechthoek te tekenen.
- `renderShader()`: in de methode `renderShader()` roept de code de methode `Graphics.beginShaderFill()` aan om een vulling met arcering op te geven. Vervolgens wordt een rechthoek getekend waarvan de vulling wordt gedefinieerd door de arceringsuitvoer (de gegenereerde kleur). Zie "Een arcering gebruiken als een vulling bij het tekenen" op pagina 329 voor meer informatie over het gebruik van een arcering op deze manier.

Hier volgt de ActionScript-code voor dit voorbeeld. Gebruik deze klasse als de hoofdtoepassingsklasse voor een alleen-ActionScript-project in Flash Builder of als de documentklasse voor het FLA-bestand in 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();
        }
    }
}
```

Hier volgt de broncode voor de ColorFilter-arceringskernel die wordt gebruikt om het Pixel Bender-bytecodebestand ColorFilter.pbj te maken:

```
<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;
    }
}
```

Als u een arcering gebruikt waarvan de parameters moeten worden gedocumenteerd, kunt u erachter komen hoeveel elementen van welk type in de array moeten worden opgenomen door de eigenschap `type` van het ShaderParameter-object te controleren. De eigenschap `type` geeft het gegevenstype van de parameter aan, zoals in de arcering zelf is gedefinieerd. Zie de eigenschappenlijst `ShaderParameter.value` in de Naslaggids voor ActionScript® 3.0 voor Adobe® Flash® Professional CS5 voor een lijst van het aantal en type elementen die door elk parametertype worden verwacht.

Elk ShaderParameter-object heeft ook een eigenschap `index` die aangeeft waar de parameter past in de volgorde van arceringsparameters. Naast deze eigenschappen kan een ShaderParameter-object extra eigenschappen hebben met metagegevenswaarden die door de auteur van de arcering worden verstrekt. De auteur kan bijvoorbeeld waarden voor metagegevens opgeven, zoals minimum-, maximum- en standaardwaarden voor een parameter. Alle metagegevenswaarden die de auteur opgeeft, worden als dynamische eigenschappen toegevoegd aan het ShaderParameter-object. Als u de eigenschappen wilt onderzoeken, gebruikt u een `for..in`-lus om de dynamische eigenschappen van het ShaderParameter-object te doorlopen. In het volgende voorbeeld wordt getoond hoe u een `for..in`-lus gebruikt om de metagegevens van een ShaderParameter-object te identificeren. Elke metagegevenswaarde wordt toegevoegd aan een Vector-instantie met de naam `metadata`. In dit voorbeeld wordt aangenomen dat er al een Shader-instantie met de naam `myShader` is gemaakt en dat deze instantie een parameter heeft met de naam `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
```

Een arcering gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als er een Pixel Bender-arcering beschikbaar is als Shader-object in ActionScript, kan de arcering op verschillende manieren worden gebruikt:

- Vulling bij het tekenen van een arcering: in de arcering wordt gedefinieerd welk gedeelte wordt gevuld van een vorm die met de teken-api is getekend.
- Overvloeimodus: de arcering bepaalt de overvloeiing tussen twee elkaar overlappende weergaveobjecten.
- Filter: in de arcering wordt een filter gedefinieerd waarmee het uiterlijk van visuele inhoud wordt gewijzigd.
- Zelfstandige verwerking van arcering: de verwerking van de arcering wordt uitgevoerd; er wordt niet opgegeven waarvoor de uitvoer wordt gebruikt. De arcering kan eventueel ook op de achtergrond worden uitgevoerd. Het resultaat is beschikbaar zodra de verwerking is voltooid. Deze techniek kan worden gebruikt om bitmapgegevens te genereren en ook om niet-visuele gegevens te verwerken.

Een arcering gebruiken als een vulling bij het tekenen

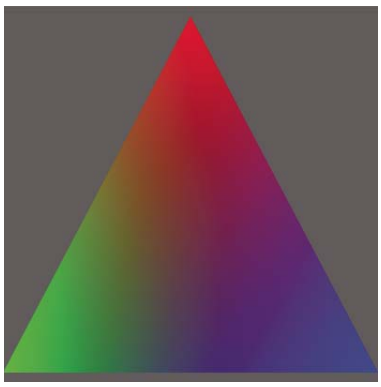
Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als u een arcering gebruikt om een vulling te maken bij het tekenen, gebruikt u de methoden van de API voor tekenen om een vectorvorm te maken. De uitvoer van de arcering wordt gebruikt om de vorm te vullen, op dezelfde manier dat een bitmapafbeelding kan worden gebruikt als bitmapvulling met de API voor tekenen. U maakt een vulling met arcering door op het punt in de code waar u wilt beginnen met het tekenen van de vorm, de methode `beginShaderFill()` van het Graphics-object op te roepen. Geef het Shader-object als het eerste argument door aan de methode `beginShaderFill()`, zoals in het codevoorbeeld wordt geïllustreerd:

```
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
```

Als u een arcering gebruikt als vulling bij het tekenen, stelt u alle waarden voor de invoerafbeelding en parameters in die de arcering nodig heeft.

In het volgende voorbeeld ziet u hoe u een arcering gebruikt als vulling bij het tekenen. In dit voorbeeld maakt de arcering een verloop met drie punten. Het verloop heeft drie kleuren, elk op het punt van een driehoek, met een overvloeiing ertussen. Daarnaast draaien de kleuren om een bewegend effect van draaiende kleuren te creëren.



Opmerking: *De code voor dit voorbeeld is geschreven door Petri Leskinen. Onze dank gaat uit naar Petri, die dit voorbeeld met ons deelt. U kunt meer voorbeelden en zelfstudies van Petri zien op http://pixelero.wordpress.com/.*

De ActionScript-code heeft drie methoden:

- `init()`: de methode `init()` wordt aangeroepen wanneer de toepassing wordt geladen. In deze methode stelt de code de beginwaarden in voor de Point-objecten die de punten van de driehoek weergeven. De code maakt ook de nieuwe Sprite-instantie `canvas`. Later, in de methode `updateShaderFill()`, tekent de code het resultaat van de arcering één keer per frame in `canvas`. Tot slot laadt de code het bytecodebestand voor de arcering.
- `onLoadComplete()`: in de methode `onLoadComplete()` maakt de code het Shader-object `shader`. De code stelt ook de beginwaarden van de parameters in. Tot slot voegt de code de methode `updateShaderFill()` toe als listener voor de gebeurtenis `enterFrame`. De methode wordt dan één keer per frame aangeroepen om een animatie-effect te maken.

- `updateShaderFill()`: de methode `updateShaderFill()` wordt één keer per frame aangeroepen om het animatie-effect te maken. In deze methode worden de parameterwaarden voor de arcering berekend en ingesteld. De code roept vervolgens de methode `beginShaderFill()` aan om een vulling met arcering te maken en roept andere methoden van de API voor tekenen aan om het resultaat van de arcering te tekenen in een driehoek.

Hier volgt de ActionScript-code voor dit voorbeeld. Gebruik deze klasse als de hoofdtoepassingsklasse voor een alleen-ActionScript-project in Flash Builder of als de documentklasse voor het FLA-bestand in 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();
        }
    }
}
```

Hier volgt de broncode voor de ThreePointGradient-arceringskernel die wordt gebruikt om het Pixel Bender-bytecodebestand ThreePointGradient.pbj te maken:

```
<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);
    }
}
```

Opmerking: *Wanneer u een arceringsvulling gebruikt wanneer u rendering uitvoert onder de GPU, wordt het gevulde gebied cyaan.*

Zie “De teken-API gebruiken” op pagina 235 voor meer informatie over het tekenen van vormen met de API voor tekenen.

Arceringen als overvloeimodus gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het gebruik van een arcering als overvloeimodus is vergelijkbaar met andere overvloeimodi. De arcering definieert de weergave die het resultaat is van twee weergaveobjecten die visueel in elkaar overvloeien. Als u een arcering als overvloeimodus wilt gebruiken, wijst u het Shader-object toe aan de eigenschap `blendShader` van het weergaveobject op de voorgrond. Door een andere waarde dan `null` toe te wijzen aan de eigenschap `blendShader`, wordt de eigenschap `blendMode` van het weergaveobject automatisch ingesteld op `BlendMode.SHADER`. In het volgende codevoorbeeld ziet u hoe u een arcering gebruikt als overvloeimodus. In dit voorbeeld wordt verondersteld dat er een weergaveobject met de naam `foreground` is in hetzelfde bovenliggende item in de weergavelijst als andere weergave-inhoud, waarbij `foreground` de andere inhoud overlapt:

```
foreground.blendShader = myShader;
```

Wanneer u een arcering als overvloeimodus gebruikt, moet de arcering zijn gedefinieerd met ten minste twee invoeren. Zoals u in het voorbeeld ziet, stelt u de invoerwaarden niet in de code in. In plaats daarvan worden de twee overgevloeide afbeeldingen automatisch gebruikt als invoer voor de arcering. De voorgrondafbeelding wordt ingesteld als de tweede afbeelding. Dit is het weergaveobject waarop de overvloeimodus wordt toegepast. Er wordt een achtergrondafbeelding gemaakt door de combinatie te nemen van alle pixels achter het kader van de voorgrondafbeelding. De achtergrondafbeelding wordt ingesteld als de eerste invoerafbeelding. Als u een arcering gebruikt die meer dan twee invoeren verwacht, geeft u voor elke invoer na de eerste twee een waarde op.

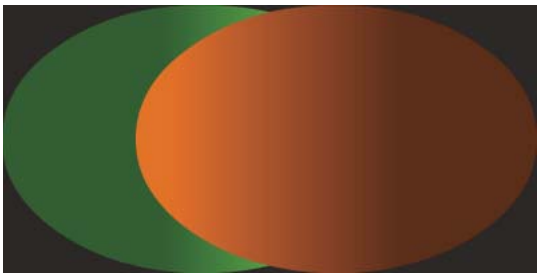
In het volgende codevoorbeeld ziet u hoe u een arcering als overvloeimodus gebruikt. In dit voorbeeld wordt een lichtermakende overvloeimodus op basis van lichtsterkte gebruikt. Het resultaat van de overvloeiing is dat de lichtste pixelwaarde van de overgevloeide objecten, de pixel is die wordt weergegeven.

Opmerking: *De code voor dit voorbeeld is geschreven door Mario Klingemann. Onze dank gaat uit naar Mario, die dit voorbeeld met ons deelt. U kunt meer werk van Ryan bekijken en zijn codes lezen op www.quasimondo.com/.*

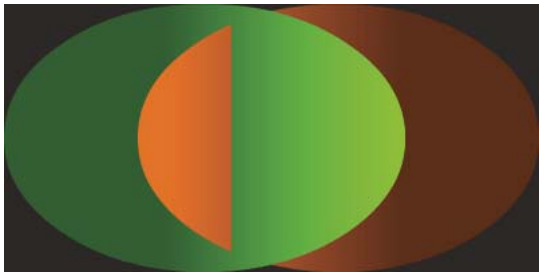
De belangrijke ActionScript-code heeft deze twee methoden:

- `init()`: de methode `init()` wordt aangeroepen wanneer de toepassing wordt geladen. In deze methode laadt de code het bytecodebestand voor de arcering.
- `onLoadComplete()`: in de methode `onLoadComplete()` maakt de code het Shader-object `shader`. Vervolgens worden drie objecten getekend. Het eerste, `backdrop`, is een donkergrijze achtergrond achter de overgevloeide objecten. Het tweede, `backgroundShape`, is een ellips met een groen kleurverloop. Het derde, `foregroundShape`, is een ellips met een oranje kleurverloop.

  De ellips `foregroundShape` is het voorgrondobject van de overvloeiing. De achtergrondafbeelding van de overvloeiing wordt gevormd door het deel van `backdrop` en het deel van `backgroundShape` die worden overlapt door het kader van het `foregroundShape`-object. Het `foregroundShape`-object is het object helemaal vooraan in de weergavelijst. Het overlapt `backgroundShape` gedeeltelijk en overlapt `backdrop` volledig. Vanwege deze overlapping zou, als er geen overvloeimodus werd toegepast, de oranje ellips (`foregroundShape`) volledig zichtbaar zijn en zou een gedeelte van de groene ellips (`backgroundShape`) er door worden verborgen:



Maar als de overvloeimodus wordt toegepast, schijnt het lichtere deel van de groene ellips als het ware door de oranje ellips omdat dit deel lichter is dan het deel van `foregroundShape` dat het overlapt:



Hier volgt de ActionScript-code voor dit voorbeeld. Gebruik deze klasse als de hoofdtoepassingsklasse voor een alleen-ActionScript-project in Flash Builder of als de documentklasse voor het FLA-bestand in 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;
        }
    }
}
```

Hier volgt de broncode voor de LumaLighten-arceringskernel die wordt gebruikt om het Pixel Bender-bytecodebestand LumaLighten.pbj te maken:

```
<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;
    }
}
```

Zie "Overvloeimodi toepassen" op pagina 197 voor meer informatie over het gebruik van overvloeimodi.

Opmerking: *Wanneer een Pixel Bender-arceringsprogramma wordt uitgevoerd als arceringsfunctie in Flash Player of AIR, gedragen de arceringsfunctie en de functie* `outCoord()` *zich anders dan in andere contexten. Bij een arcering retourneert de samplingfunctie altijd de huidige pixel die door de arceringsfunctie wordt geëvalueerd. Het is bijvoorbeeld niet mogelijk om een verschuiving te gebruiken bij* `outCoord()` *om een nabijgelegen pixel te evalueren. Op dezelfde manier geldt dat als u de functie* `outCoord()` *buiten een samplingfunctie gebruikt, dat de coördinaten altijd worden geëvalueerd als 0. U kunt bijvoorbeeld niet de positie van een pixel gebruiken om de manier waarop de gearceerde afbeeldingen worden gecombineerd te beïnvloeden.*

Arceringen als filter gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het gebruik van een arcering als filter is vergelijkbaar met het gebruik van andere filters in ActionScript. Als u een arcering als filter gebruikt, wordt de gefilterde afbeelding (een weergave- of BitmapData-object) doorgegeven aan de arcering. De arcering gebruikt de invoerafbeelding om de filteruitvoer te maken. Dit is doorgaans een gewijzigde versie van de oorspronkelijke afbeelding. Als het gefilterde object een weergaveobject is, wordt de uitvoer van de arcering op het scherm weergegeven in plaats van het gefilterde weergaveobject. Als het gefilterde object een BitmapData-object is, wordt de uitvoer van de arcering de inhoud van het BitmapData-object waarvan de methode `applyFilter()` is aangeroepen.

Wanneer u een arcering als filter wilt gebruiken, maakt u eerst het Shader-object zoals is beschreven in "Arceringen laden of insluiten" op pagina 320. Vervolgens maakt u een ShaderFilter-object dat is gekoppeld aan het Shader-object. Het ShaderFilter-object is het filter dat wordt toegepast op het gefilterde object. U kunt het op dezelfde manier als een ander filter toepassen op een object. U geeft het filter door aan de eigenschap `filters` van een weergaveobject of u roept de methode `applyFilter()` aan voor een BitmapData-object. Met de volgende voorbeeldcode wordt een ShaderFilter-object gemaakt en het filter toegepast op het weergaveobject `homeButton`:

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

Wanneer u een arcering als filter gebruikt, moet de arcering met ten minste één invoer zijn gedefinieerd. Zoals u in het voorbeeld ziet, stelt u de invoerwaarde niet in de code in. In plaats daarvan wordt het gefilterde weergave- of BitmapData-object ingesteld als invoerafbeelding. Als u een arcering gebruikt die meer dan één invoer verwacht, geeft u voor elke invoer na de eerste invoer een waarde op.

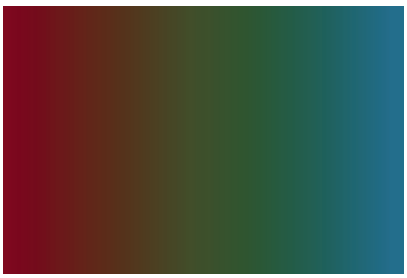
In sommige gevallen verandert een filter de afmetingen van de oorspronkelijke afbeelding. Een slagschaduweffect voegt bijvoorbeeld extra pixels toe aan de schaduw die is toegevoegd aan de afbeelding. Wanneer u een arcering gebruikt die de afmetingen van de afbeelding wijzigt, stelt u de eigenschappen `leftExtension`, `rightExtension`, `topExtension` en `bottomExtension` in om aan te geven in welke mate u de grootte van de afbeelding wilt veranderen.

In het volgende voorbeeld ziet u hoe u een arcering als filter gebruikt. Het filter in dit voorbeeld keert de waarden voor het rode, groene en blauwe kanaal van een afbeelding om. Het resultaat is de "negatieve" versie van de afbeelding.

Opmerking: *De arcering die in dit voorbeeld wordt gebruikt, is de Pixel Bender-kernel invertRGB.pbk die deel uitmaakt van de Pixel Bender Toolkit. U kunt de broncode voor de kernel laden vanuit de installatiemap van de Pixel Bender Toolkit. Compileer de broncode en sla het bestand met bytecode op in dezelfde map als de broncode.*

De belangrijke ActionScript-code heeft deze twee methoden:

- `init()`: de methode `init()` wordt aangeroepen wanneer de toepassing wordt geladen. In deze methode laadt de code het bytecodebestand voor de arcering.
- `onLoadComplete()`: in de methode `onLoadComplete()` maakt de code het Shader-object `shader`. Vervolgens wordt de inhoud van het object `target` gemaakt en getekend. Het object `target` is een rechthoek gevuld met een lineair kleurverloop dat rood is aan de linkerzijde, geel-groen in het midden en lichtblauw aan de rechterzijde. Het ongefilterde object ziet er zo uit:



Nadat het filter is toegepast, zijn de kleuren omgekeerd en ziet de rechthoek er zo uit:

De arcering die in dit voorbeeld wordt gebruikt, is de Pixel Bender-kernel invertRGB.pbk die deel uitmaakt van de Pixel Bender Toolkit. De broncode is beschikbaar in het bestand invertRGB.pbk in de installatiemap van de Pixel Bender Toolkit. Compileer de broncode en sla het bytecodebestand op onder de naam invertRGB.pbj in dezelfde map als de ActionScript-broncode.

Hier volgt de ActionScript-code voor dit voorbeeld. Gebruik deze klasse als de hoofdtoepassingsklasse voor een alleen-ActionScript-project in Flash Builder of als de documentklasse voor het FLA-bestand in 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];
        }
    }
}
```

Zie "Filters maken en toepassen" op pagina 284 voor meer informatie over het toepassen van filters.

Arceringen in zelfstandige modus gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als u een arcering in zelfstandige modus gebruikt, wordt de arcering uitgevoerd onafhankelijk van het gebruik dat u wilt maken van de uitvoer. U geeft op welke arcering moet worden uitgevoerd, stelt invoer- en parameterwaarden in en geeft op in welk object de resultaatgegevens moeten worden geplaatst. Er zijn twee redenen om een arcering in zelfstandige modus te gebruiken:

- Verwerken van niet-grafische gegevens: in zelfstandige modus kunt u willekeurige binaire of numerieke gegevens in plaats van grafische bitmapgegevens aan de arcering doorgeven. U kunt het resultaat van de arcering niet alleen als grafische bitmapgegevens maar ook als binaire of numerieke gegevens laten retourneren.
- Achtergrondverwerking: wanneer u een arcering in zelfstandige modus uitvoert, wordt de arcering standaard asynchroon uitgevoerd. Dit betekent dat de arcering op de achtergrond wordt uitgevoerd terwijl uw toepassing blijft lopen. Wanneer de arceringsbewerking is voltooid, ontvangt de code een melding. U kunt een arcering gebruiken waarvan de uitvoering lang duurt, zonder dat de gebruikersinterface van de toepassing bevriest of andere processen worden stilgezet terwijl de arcering wordt uitgevoerd.

U kunt een ShaderJob-object gebruiken om een arcering in zelfstandige modus uit te voeren. Eerst maakt u het ShaderJob-object en koppelt u dit aan het Shader-object dat de arcering vertegenwoordigt die moet worden uitgevoerd:

```
var job:ShaderJob = new ShaderJob(myShader);
```

Vervolgens stelt u de invoer- of parameterwaarden in die de arcering verwacht. Als u de arcering op de achtergrond uitvoert, kunt u ook een listener registreren voor de gebeurtenis `complete` van het ShaderJob-object. De listener wordt opgeroepen wanneer de arcering het werk heeft voltooid:

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

Vervolgens maakt u een object waarin het resultaat van de arceringsbewerking wordt geschreven nadat de bewerking is voltooid. U wijst dat object toe aan de eigenschap `target` van het ShaderJob-object:

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

Wijs een BitmapData-instantie aan de eigenschap `target` toe als u de ShaderJob gebruikt om grafische gegevens te bewerken. Als u binaire of numerieke gegevens verwerkt, wijst u een ByteArray-object of Vector.<Number>-instantie aan de eigenschap `target` toe. In dat geval moet u de eigenschappen `width` en `height` van het ShaderJob-object instellen om de hoeveelheid gegevens op te geven die naar het `target`-object wordt uitgevoerd.

Opmerking: *U kunt de eigenschappen `shader`, `target`,`width` en `height` van het ShaderJob-object in één stap instellen door argumenten door te geven aan de constructor `ShaderJob()`. U doet dit als volgt: `var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);`*

Wanneer u gereed bent om de arcering uit te voeren, roept u de methode `start()` van het ShaderJob-object aan:

```
job.start();
```

Standaard heeft het aanroepen van `start()` tot gevolg dat ShaderJob asynchroon wordt uitgevoerd. In dit geval gaat de uitvoering van het programma direct door met de volgende coderegel en wordt niet gewacht tot de arcering is voltooid. Wanneer de arceringsbewerking is voltooid, roept het ShaderJob-object de listeners voor de gebeurtenis `complete` aan om ze te melden dat de bewerking is voltooid. Op dat punt (dat wil zeggen in de hoofdtekst van de gebeurtenislistener `complete`) bevat het `target`-object het resultaat van de arceringsbewerking.

Opmerking: *In plaats van de eigenschap `target` van het object te gebruiken, kunt u het resultaat van de arcering direct ophalen uit het gebeurtenisobject dat wordt doorgegeven aan de listenermethode. Het gebeurtenisobject is een ShaderEvent-instantie. Het ShaderEvent-object heeft drie eigenschappen die kunnen worden gebruikt om het resultaat te benaderen, afhankelijk van het gegevenstype van het object dat u instelt als de eigenschap `target`: `ShaderEvent.bitmapData`, `ShaderEvent.byteArray` en `ShaderEvent.vector`.*

U kunt ook het argument `true` doorgeven aan de methode `start()`. In dat geval wordt de arcering synchroon uitgevoerd. Alle code (inclusief interactie met de gebruikersinterface en eventuele andere gebeurtenissen) wordt stopgezet terwijl de arcering wordt uitgevoerd. Wanneer de arcering is voltooid, bevat het `target`-object het resultaat van de arcering en gaat het programma verder met de volgende coderegel.

```
job.start(true);
```

Hoofdstuk 16: Werken met filmclips

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De MovieClip-klasse is de hoofdklasse voor animatie en filmclipsymbolen die u maakt in uw Adobe® Flash®-ontwikkelingsomgeving. Deze heeft het gedrag en de functionaliteit van weergaveobjecten, maar ook aanvullende eigenschappen en methoden voor het beheren van de tijdlijn.

Basisbeginselen van filmclips

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Filmclips vormen een belangrijk element voor mensen die bewegende inhoud met het Flash-ontwerpgereedschap maken en de inhoud met ActionScript willen beheren. Telkens wanneer u een filmclipsymbool in Flash maakt, voegt Flash het symbool aan de bibliotheek van dat Flash-document toe. Dit symbool wordt standaard een instantie van de MovieClip-klasse, en heeft als zodanig de eigenschappen en methoden van de MovieClip-klasse.

Wanneer een instantie van een filmclipsymbool in het werkgebied wordt geplaatst, doorloopt de filmclip automatisch de tijdlijn (als de clip uit meer dan een frame bestaat), tenzij het afspelen met behulp van ActionScript wordt gewijzigd. De klasse MovieClip wordt door deze tijdlijn onderscheiden van andere klassen. Dankzij de tijdlijn kunt u met behulp van het Flash-ontwerpgereedschap animatie maken via bewegings- of vorm-tweens. Met een weergaveobject dat een instantie van de klasse Sprite is, kunt u daarentegen alleen animatie maken door de waarden van het object via een programmacode te wijzigen.

In eerdere versies van ActionScript was de basisklasse van alle instanties in het werkgebied de klasse MovieClip. In ActionScript 3.0 is een filmclip slechts een van de vele weergaveobjecten die op het scherm kunnen worden weergegeven. Als er voor de functie van een weergaveobject geen tijdlijn nodig is, kunt u de renderprestaties verbeteren door de klasse Shape of Sprite in plaats van MovieClip te gebruiken. Zie "Een subklasse DisplayObject kiezen" op pagina 183 voor meer informatie over het kiezen van het relevante weergaveobject voor een taak.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot filmclips:

AVM1 SWF een SWF-bestand dat met behulp van ActionScript 1.0 of ActionScript 2.0 is gemaakt, meestal voor Flash Player 8 of eerder.

AVM2 SWF een SWF-bestand dat is gemaakt met ActionScript 3.0 voor Adobe Flash Player 9 of hoger, of Adobe AIR.

Externe SWF Een SWF-bestand dat gescheiden van het SWF-projectbestand is gemaakt en is bedoeld om in het SWF-projectbestand te worden geladen en binnen dat SFW-bestand te worden afgespeeld.

Frame de kleinste tijdeenheid in de tijdlijn. Zoals bij een filmstrip van een speelfilm, is elk frame een momentopname van de animatie op een bepaald moment. Wanneer frames snel achter elkaar worden afgespeeld, wordt het animatie-effect gecreëerd.

Tijdlijn De metaforische voorstelling van de serie frames, waaruit de animatiereeks van een filmclip bestaat. de tijdlijn van een MovieClip-object is gelijk aan de tijdlijn in het Flash-ontwerpgereedschap.

Afspeelkop een markering waarmee de locatie (het frame) in de tijdlijn wordt geïdentificeerd, die op dat moment wordt weergegeven.

Werken met MovieClip-objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een SWF-bestand publiceert, zet Flash alle instanties van filmclipsymbolen in het werkgebied om in MovieClip-objecten. U kunt een filmclipsymbool beschikbaar maken voor ActionScript door er in het veld Instantienaam van Eigenschapcontrole een instantienaam aan te geven. Wanneer het SWF-bestand is gemaakt, genereert Flash de code waarmee de MovieClip-instantie in het werkgebied wordt gemaakt en declareert een variabele met behulp van die instantienaam. Als u een naam geeft aan filmclips die binnen andere filmclips met een naam zijn genest, worden deze onderliggende filmclips behandeld als eigenschappen van de bovenliggende filmclip: u kunt de onderliggende filmclip met behulp van puntnotatie openen. Als een filmclip met de instantienaam `childClip` bijvoorbeeld is genest binnen een andere clip met de instantienaam `parentClip`, kunt u de tijdlijnanimatie van de onderliggende clip afspelen via de volgende code:

```
parentClip.childClip.play();
```

***Opmerking:** : Onderliggende instanties die in het werkgebied in het Flash-ontwerpgereedschap zijn geplaatst, kunnen niet worden benaderd door code vanuit de constructor van een bovenliggende instantie, omdat deze niet op dat punt in de code-uitvoering zijn gemaakt. Voordat de onderliggende instantie wordt benaderd, moet de bovenliggende instantie de onderliggende instantie met behulp van code maken of toegang vertragen tot een callback-functie die luistert naar het verzenden van de gebeurtenis `Event.ADDED_TO_STAGE` door de onderliggende instantie.*

Hoewel sommige oudere methoden en eigenschappen van de klasse MovieClip van ActionScript 2.0 hetzelfde blijven, zijn andere methoden en eigenschappen gewijzigd. Alle eigenschappen die met een onderstrepingsteken beginnen, hebben een nieuwe naam gekregen. De eigenschappen `_width` en `_height` zijn nu toegankelijk als `width` en `height`, terwijl `_xscale` en `_yscale` nu toegankelijk zijn als`scaleX` en `scaleY`. Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor een complete lijst van de eigenschappen en methoden van de klasse MovieClip.

Afspelen van filmclips besturen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash gebruikt de metafoor van een tijdlijn om animatie of wijzigingen in een toestand over te brengen. Alle visuele elementen die van een tijdlijn gebruikmaken, moeten een object MovieClip zijn of een uitbreiding van de klasse MovieClip. Hoewel ActionScript elke filmclip kan laten stoppen, afspelen of naar een ander punt in de tijdlijn kan verplaatsen, kan ActionScript niet worden gebruikt om dynamisch een tijdlijn te maken of inhoud aan specifieke frames toe te voegen. Dit kan alleen worden gedaan via het Flash-ontwerpgereedschap.

Wanneer een filmclip wordt afgespeeld, beweegt deze in de tijdlijn met een snelheid die door de framesnelheid van het SWF-bestand wordt bepaald. U kunt deze instelling ook overschrijven door de eigenschap `Stage.frameRate` in ActionScript te configureren.

Filmclips afspelen en het afspelen stoppen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methoden `play()` en `stop()` bieden basisbesturingselementen voor een filmclip in de tijdlijn. U hebt bijvoorbeeld een filmclipsymbool in het werkgebied waarin een animatie van een over het scherm bewegende fiets is opgenomen. De instantienaam is ingesteld op `bicycle` (fiets). Als de volgende code aan een hoofdframe in de hoofdtijdlijn is gekoppeld,

```
bicycle.stop();
```

beweegt de fiets niet (de animatie wordt niet afgespeeld). De beweging van de fiets kan worden gestart via een andere gebruikersinteractie. Als er bijvoorbeeld een knop met de naam `startButton` aanwezig is, zorgt de volgende code voor een hoofdframe in de hoofdtijdlijn ervoor dat de animatie door het klikken op de knop wordt afgespeeld:

```
// 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);
```

Snel vooruitspoelen en terugspoelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methoden `play()` en `stop()` vormen niet de enige manier om het afspelen van een filmclip te beheren. U kunt de afspeelkop ook handmatig vooruit of achteruit in de tijdlijn verplaatsen met behulp van de methoden `nextFrame()` en `prevFrame()`. Door het aanroepen van deze methoden wordt het afspelen gestopt en wordt de afspeelkop respectievelijk een frame vooruit of achteruit verplaatst.

Het gebruik van de methode `play()` is gelijk aan het aanroepen van `nextFrame()`, telkens wanneer de gebeurtenis `enterFrame` van het filmclipobject wordt geactiveerd. Op dezelfde manier kunt u de filmclip `bicycle` achterwaarts afspelen door een gebeurtenislistener voor de gebeurtenis `enterFrame` te maken en `bicycle` in de listenerfunctie opdracht te geven naar het vorige frame te gaan. U doet dit als volgt:

```
// 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);
```

Als een filmclip bij normaal afspelen meer dan een frame bevat, wordt de clip in een doorlopende lus afgespeeld. Dat wil zeggen dat de clip na het passeren van het laatste frame naar frame 1 terugkeert. Wanneer u `prevFrame()` of `nextFrame()` gebruikt, gebeurt dit niet automatisch (als u `prevFrame()` aanroept wanneer de afspeelkop zich op frame 1 bevindt, wordt de afspeelkop niet naar het laatste frame verplaatst). De voorwaarde `if` in het bovenstaande voorbeeld controleert of de afspeelkop achterwaarts naar het eerste frame is verplaatst en stelt de afspeelkop voor het laatste frame in. Daarmee wordt een doorlopende lus gemaakt, waarin de filmclip achterwaarts wordt afgespeeld.

Naar een ander frame springen en framelabels gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het is bijzonder eenvoudig om een filmclip naar een nieuw frame te sturen. U roept `gotoAndPlay()` of `gotoAndStop()` aan. De filmclip springt dan naar het framenummer dat als parameter is opgegeven. U kunt ook een tekenreeks doorgeven, die overeenkomt met de naam van een framelabel. U kunt een label aan alle frames in de tijdlijn toewijzen. U doet dit door een frame in de tijdlijn te selecteren en een naam in het veld Framelabel van Eigenschapcontrole in te vullen.

De voordelen van het gebruik van framelabels in plaats van nummers worden vooral duidelijk wanneer u een ingewikkelde filmclip maakt. Wanneer het aantal frames, lagen en tweens in een animatie groot wordt, kunt u belangrijke frames een label geven met verklarende beschrijvingen over ander gedrag van de filmclip (bijvoorbeeld 'uit', 'lopen' of 'rennen'). Hierdoor wordt de leesbaarheid van de code verhoogd. Deze methode biedt tevens flexibiliteit, aangezien ActionScript-aanroepen die naar een frame met een label gaan, naar een enkele referentie verwijzen (het label), in plaats van naar een specifiek framenummer. Als u later besluit een bepaald segment van de animatie naar een ander frame te verplaatsen, hoeft u de ActionScript-code niet te wijzigen, als u voor de frames in de nieuwe locatie tenminste hetzelfde label gebruikt.

ActionScript 3.0 bevat de klasse FrameLabel voor het vertegenwoordigen van framelabels in code. Elke instantie van deze klasse vertegenwoordigt een enkel framelabel en heeft een eigenschap `name` waarmee de naam van het framelabel volgens de specificatie in Eigenschapcontrole wordt aangegeven, en een eigenschap `frame` waarmee het framenummer (de plaats van het frame in de tijdlijn) wordt aangegeven.

Voor toegang tot FrameLabel-instanties die aan een filmclipinstantie zijn gekoppeld, bevat de klasse MovieClip twee eigenschappen waarmee direct objecten FrameLabel worden geretourneerd. De eigenschap `currentLabels` retourneert een array dat uit alle objecten FrameLabel in de volledige tijdlijn van een filmclip bestaat. De eigenschap `currentLabel` retourneert een reeks met de naam van het framelabel dat het laatst op de tijdlijn is aangetroffen.

Stel dat u een filmclip met de naam `robot` maakt, waarbij u een label aan de verschillende toestanden van de animatie geeft. U kunt een voorwaarde instellen, waarmee de eigenschap `currentLabel` wordt gecontroleerd op toegang tot de huidige toestand van `robot`, zoals in de volgende code:

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

In Flash Player 11.3 en AIR 3.3 is de gebeurtenis `frameLabel` toegevoegd aan de FrameLabel-klasse. U kunt een gebeurtenishandler toewijzen aan de FrameLabel-instantie die een framelabel vertegenwoordigt. De gebeurtenis wordt verzonden wanneer de afspeelkop het frame betreedt.

In het volgende voorbeeld wordt een FrameLabel-instantie gemaakt voor het tweede framelabel in de array met framelabels voor de MovieClip. Vervolgens wordt een gebeurtenishandler geregistreerd voor de gebeurtenis `frameLabel`:

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

Werken met scènes

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de Flash-ontwerpomgeving kunt u scènes gebruiken voor het omsluiten van een serie tijdlijnen die een SWF-bestand doorloopt. Als u de tweede parameter van de methoden `gotoAndPlay()` of `gotoAndStop()` gebruikt, kunt u opgeven naar welke scène de afspeelkop wordt verplaatst. Alle FLA-bestanden beginnen met alleen de eerste scène, maar u kunt ook nieuwe scènes maken.

Het gebruik van scènes is niet altijd de beste oplossing, omdat scènes een aantal nadelen hebben. Een Flash-document dat meerdere scènes bevat, kan moeilijk te onderhouden zijn, met name in omgevingen met meerdere ontwikkelaars. Meerdere scènes zijn ook niet erg efficiënt qua bandbreedte, omdat alle scènes door het publicatieproces in een enkele tijdlijn worden samengevoegd. Hierdoor worden alle scènes progressief gedownload, zelfs als ze nooit worden afgespeeld. Daarom wordt het gebruik van meerdere scènes vaak afgeraden, behalve bij het ordenen van lange animaties die op meerdere tijdlijnen zijn gebaseerd.

De eigenschap `scenes` van de klasse MovieClip retourneert een array Scene-objecten die alle scènes van het SWF-bestand vertegenwoordigen. De eigenschap `currentScene` retourneert een object Scene dat de op dat moment afgespeelde scène vertegenwoordigt.

De klasse Scene biedt verschillende eigenschappen die informatie over een scène geven. De eigenschap `labels` retourneert een array FrameLabel-objecten die de framelabels in die scène vertegenwoordigen. De eigenschap `name` retourneert de naam van de scène als tekenreeks. De eigenschap `numFrames` retourneert een geheel getal dat het totale aantal frames in de scène vertegenwoordigt.

MovieClip-objecten met ActionScript maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een manier om in Flash inhoud aan het scherm toe te voegen, is het slepen van elementen van de bibliotheek naar het werkgebied. Dit is echter niet het enige werkschema. Bij ingewikkelde projecten geven ervaren ontwikkelaars vaak de voorkeur aan het maken van filmclips via programmacodes. Deze benadering biedt verschillende voordelen: eenvoudig hergebruik van code, snellere compilatie en geavanceerde wijzigingen die alleen in ActionScript beschikbaar zijn.

De weergaveoverzicht-API van ActionScript 3.0 stroomlijnt het proces waarbij dynamisch MovieClip-objecten worden gemaakt. De mogelijkheid om een MovieClip-instantie rechtstreeks te instantiëren, los van het proces waarbij de instantie aan het weergaveoverzicht wordt toegevoegd, biedt flexibiliteit en eenvoud zonder de beheermogelijkheden te beperken.

Wanneer u in ActionScript 3.0 via programmacode een filmclipinstantie (of een ander weergaveobject) maakt, is de instantie pas op het scherm zichtbaar wanneer deze aan het weergaveoverzicht is toegevoegd door de methode `addChild()` of `addChildAt()` op een weergaveobjectcontainer aan te roepen. Hierdoor kunt u een filmclip maken, de eigenschappen ervan instellen en zelfs methoden aanroepen, voordat de filmclip naar het scherm wordt gerenderd. Zie “Werken met weergaveobjectcontainers” op pagina 169 voor meer informatie over het werken met de weergavelijst.

Bibliotheeksymbolen voor ActionScript exporteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Instanties van filmclipsymbolen in de bibliotheek van een Flash-document kunnen standaard niet dynamisch worden gemaakt (dat wil zeggen gemaakt met alleen ActionScript). De reden hiervoor is dat elk symbool dat voor gebruik in ActionScript wordt geëxporteerd, uw SWF-bestand groter maakt en het bekend is dat sommige symbolen niet zijn bedoeld voor gebruik in het werkgebied. Daarom moet u, als u een symbool in ActionScript beschikbaar wilt maken, opgeven dat het symbool voor ActionScript moet worden geëxporteerd.

U kunt als volgt een symbool voor ActionScript exporteren:

1. Selecteer het symbool in het deelvenster Bibliotheek en open het dialoogvenster met de symbooleigenschappen.
2. Activeer zo nodig de Geavanceerde instellingen.
3. Schakel in het gedeelte Koppeling het selectievakje Exporteren voor ActionScript in.

   De velden Klasse en Basisklasse worden nu geactiveerd.

   Het veld Klasse bevat standaard de symboolnaam, zonder spaties (een symbool met de naam “Hoog Huis" wordt bijvoorbeeld “HoogHuis”). Als u wilt opgeven dat het symbool een aangepaste klasse voor het gedrag moet gebruiken, vult u de volledige naam van de klasse, inclusief het pakket ervan, in. Als u instanties van het symbool in ActionScript wilt maken, maar geen aanvullend gedrag wilt toevoegen, hoeft u de naam van de klasse niet te wijzigen.

   De waarde van het veld Basisklasse is standaard `flash.display.MovieClip`. Als u wilt dat uw symbool met de functionaliteit van een andere klantenklasse wordt uitgebreid, kunt u in plaats daarvan de naam van die klasse opgeven, als die klasse tenminste een uitbreiding van de klasse Sprite (of MovieClip) is.
4. Klik op OK om uw wijzigingen op te slaan.

   Als Flash op dit punt geen gekoppeld SWC-bestand of geen extern ActionScript-bestand met een definitie voor de opgegeven klasse kan vinden (als u bijvoorbeeld geen aanvullend gedrag voor het symbool hoefde toe te voegen), wordt een waarschuwing weergegeven:

   Kan geen definitie voor deze klasse vinden in het klassenpad. Er wordt automatisch een definitie gegenereerd in het SWF-bestand bij exporteren.

   U kunt deze waarschuwing negeren als uw bibliotheeksymbool naast de functionaliteit van de klasse MovieClip geen unieke functionaliteit nodig heeft.

Als u geen klasse voor uw symbool opgeeft, maakt Flash een klasse voor uw symbool, die gelijk is aan dit symbool:

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

Als u wel extra ActionScript-functionaliteit aan uw symbool wilt toevoegen, voegt u de relevante eigenschappen en methoden aan de codestructuur hieronder toe. Stel dat u bijvoorbeeld een filmclipsymbool hebt dat een cirkel van 50 pixels breed en 50 pixels hoog bevat en dat is opgegeven dat het voor ActionScript wordt geëxporteerd met een klasse met de naam Circle. Als de volgende code in een bestand Circle.as wordt geplaatst, vormt de code een uitbreiding van de klasse MovieClip. Het symbool heeft dan de aanvullende methoden `getArea()` en `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;
        }
    }
}
```

De volgende code, geplaatst op een hoofdframe op frame 1 van het Flash-document, maakt een instantie van het symbool en geeft deze op het scherm weer:

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

Deze code demonstreert het maken van een op ActionScript gebaseerde instantie in plaats van individuele elementen naar het werkgebied te slepen. Er wordt een cirkel gemaakt die alle eigenschappen van een filmclip heeft en tevens de aangepaste methoden die in de klasse Circle zijn gedefinieerd. Dit is een eenvoudig voorbeeld. Uw bibliotheeksymbool kan een willekeurige hoeveelheid eigenschappen en methoden in de klasse opgeven.

Instanties op basis van ActionScript maken is een krachtig hulpmiddel, omdat u er dynamisch grote hoeveelheden instanties mee kunt maken, die handmatig erg veel werk zouden kosten. Het biedt ook flexibiliteit, omdat u de eigenschappen van elke instantie tijdens het maken ervan kunt aanpassen. U kunt een indruk van deze twee voordelen krijgen als u een lus gebruikt om meerdere Circle-instanties dynamisch te maken. Gebruik het symbool en de klasse Circle die eerder in de Flash-documentbibliotheek zijn beschreven en plaats de volgende code op een hoofdframe op frame 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);
}
```

Dit demonstreert hoe u snel via code meerdere instanties van een symbool kunt maken en aanpassen. Elke instantie wordt op basis van het huidige getal binnen de lus geplaatst en elke instantie krijgt een willekeurige kleur door de eigenschap `transform` in te stellen (Circle overerft die eigenschap automatisch door de klasse MovieClip uit te breiden).

Een extern SWF-bestand laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

SWF-bestanden worden in ActionScript 3.0 geladen met behulp van de klasse Loader. Als u een extern SWF-bestand wilt laden, moet uw ActionScript vier dingen doen:

1. Een nieuw object URLRequest met de URL van het bestand maken.
2. Een nieuw object Loader maken.
3. De methode `load()` van het object Loader aanroepen en de URLRequest-instantie als parameter doorgeven.
4. De methode `addChild()` op een weergaveobjectcontainer (zoals de hoofdtijdlijn van een Flash-document) aanroepen om de Loader-instantie aan het weergaveoverzicht toe te voegen.

De code ziet er uiteindelijk als volgt uit:

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

Dezelfde code kan worden gebruikt om een extern beeldbestand, zoals JPEG, GIF of PNG te laden door de URL van het afbeeldingsbestand in plaats van de URL van een SWF-bestand op te geven. In tegenstelling tot afbeeldingsbestanden, kan een SWF-bestand ActionScript bevatten. Hoewel het proces voor het laden van een SWF-bestand dus hetzelfde is als het proces voor het laden van een afbeelding, moeten bij het laden van een extern SWF-bestand zowel het SWF-bestand dat het laden uitvoert als het SWF-bestand dat wordt geladen, zich in dezelfde beveiligingssandbox bevinden, als Flash Player of AIR het SWF-bestand afspelen en u van plan bent ActionScript te gebruiken om met het externe SWF-bestand te communiceren. Als het externe SWF-bestand daarnaast klassen bevat die dezelfde naamruimte delen als klassen in het ladende SWF-bestand, moet u mogelijk een nieuw toepassingsdomein voor het geladen SWF-bestand maken om naamruimteconflicten te voorkomen. Zie "Werken met toepassingsdomeinen" op pagina 157 en "Inhoud laden" op pagina 1126 voor meer informatie over overwegingen met betrekking tot veiligheid en toepassingsdomeinen.

Wanneer het laden van het externe SWF-bestand is voltooid, kunt u het bestand openen via de eigenschap `Loader.content`. Als het externe SWF-bestand voor ActionScript 3.0 is gepubliceerd, is het een filmclip of een sprite, afhankelijk van de klasse waarvan het bestand een uitbreiding is.

Er zijn een paar verschillen voor het laden van een SWF-bestand in Adobe AIR voor iOS ten opzichte van andere platforms. Zie "SWF-bestanden laden in AIR voor iOS" op pagina 214 voor meer informatie.

Overwegingen bij het laden van een ouder SWF-bestand

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als het externe SWF-bestand is gepubliceerd met een oudere versie van ActionScript, moeten er belangrijke beperkingen in overweging worden genomen. In tegenstelling tot een ActionScript 3.0 SWF-bestand dat wordt uitgevoerd in AVM2 (ActionScript Virtual Machine 2), wordt een SWF-bestand dat is gepubliceerd voor ActionScript 1.0 of 2.0 in AVM1 (ActionScript Virtual Machine 1) uitgevoerd.

Er zijn belangrijke verschillen tussen het laden van een SWF-bestand van ActionScript 1.0 en 2.0 in een SWF-bestand van ActionScript 3.0 en het laden van een SWF-bestand van ActionScript 3.0. Flash Player biedt volledige achterwaartse compabiliteit met vorige gepubliceerde inhoud. Inhoud die in vorige versies van Flash Player kan worden afgespeeld, kan ook worden uitgevoerd in versies van Flash Player die ActionScript 3.0 ondersteunen. De volgende beperkingen zijn van toepassing:

- ActionScript 3.0-code kan een SWF-bestand laden dat geschreven is in ActionScript 1.0 of 2.0. Wanneer een SWF-bestand van ActionScript 1.0 of 2.0 is geladen, is het geladen object (de eigenschap `Loader.Content`) is een AVM1Movie-object. Een AVM1Movie-instantie is niet hetzelfde als een MovieClip-instantie. Het is een weergaveobject, maar in tegenstelling tot een filmclip, bevat dit weergaveobject geen methoden of eigenschappen met betrekking tot een tijdlijn. Het bovenliggende AVM2 SWF-bestand heeft geen toegang tot de eigenschappen, methoden of objecten van het geladen AVM1-filmobject.
- SWF-bestanden die zijn geschreven in ActionScript 1.0 of 2.0 kunnen geen SWF-bestanden laden die zijn geschreven in ActionScript 3.0. Dit betekent dat SWF-bestanden die zijn gemaakt in Flash 8 of Flex Builder 1.5 of eerdere versies geen ActionScript 3.0 SWF-bestanden kunnen laden.

  De enige uitzondering op deze regel is dat een SWF-bestand van ActionScript 2.0 zichzelf door een SWF-bestand van ActionScript 3.0 kan vervangen, indien het SWF-bestand van ActionScript 2.0 niets eerder in een van zijn niveaus heeft geladen. Een SWF-bestand van ActionScript 2.0 kan dit doen door `loadMovieNum()` aan te roepen en daarbij een waarde van 0 aan de parameter `level` door te geven.

- Over het algemeen moeten SWF-bestanden, die geschreven zijn in ActionScript 1.0 of 2.0, worden gemigreerd als ze samenwerken met SWF-bestanden die geschreven zijn in ActionScript 3.0. U hebt bijvoorbeeld een mediaspeler gemaakt met ActionScript 2.0. De mediaspeler laadt verschillende soorten inhoud die is gemaakt met ActionScript 2.0. U kunt geen nieuwe inhoud maken in ActionScript 3.0 en deze dan laden in de mediaspeler. U moet de mediaspeler naar ActionScript 3.0 migreren.

  Als u echter een mediaspeler in ActionScript 3.0 maakt, kan die mediaspeler uw ActionScript 2.0-inhoud op eenvoudige wijze laden.

In de onderstaande tabellen worden de beperkingen van vorige versies van Flash Player met betrekking tot het laden van nieuwere inhoud en het uitvoeren van code weergegeven. Hier vindt u ook beperkingen voor cross-scripting tussen SWF-bestanden die geschreven zijn in verschillende versies van ActionScript.

Ondersteunde functionaliteit	**Flash Player 7**	**Flash Player 8**	**Flash Player 9 en 10**
Kan SWF-bestanden laden die zijn gepubliceerd voor	7 en eerder	8 en eerder	9 (of 10) of lager
Bevat deze AVM	AVM1	AVM1	AVM1 en AVM2
Voert SWF-bestanden uit die zijn geschreven in ActionScript	1.0 en 2.0	1.0 en 2.0	1.0, 2.0 en 3.0

In de volgende tabel duidt "Ondersteunde functionaliteit" op inhoud die in Flash Player 9 of hoger wordt uitgevoerd. Inhoud die in Flash Player 8 of eerder wordt uitgevoerd, kan alleen ActionScript 1.0 en 2.0 laden, weergeven, uitvoeren en scripts uitwisselen.

Ondersteunde functionaliteit	**Inhoud die is gemaakt in ActionScript 1.0 en 2.0**	**Inhoud die is gemaakt in ActionScript 3.0**
Kan inhoud laden en code in inhoud uitvoeren gemaakt in	inhoud die in ActionScript 1.0 en 2.0 is gemaakt	ActionScript 1.0, 2.0 en ActionScript 3.0
Cross-scripting is uitsluitend mogelijk met	Alleen ActionScript 1.0 en 2.0 (ActionScript 3.0 via LocalConnection)	ActionScript 1.0 en 2.0 via LocalConnection. ActionScript 3.0

Voorbeeld van een filmclip: RuntimeAssetsExplorer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De functie Exporteren voor ActionScript kan vooral voordelen hebben voor bibliotheken die in meer dan een project worden gebruikt. Als Flash Player of AIR een SWF-bestand uitvoert, zijn symbolen die naar ActionScript zijn geëxporteerd, beschikbaar voor elk SWF-bestand in dezelfde beveiligingssandbox als het SWF-bestand dat het laadt. Zo kan één Flash-document een SWF-bestand genereren dat alleen is bestemd voor het bevatten van grafische elementen. Deze techniek is vooral handig voor grotere projecten, waarbij ontwikkelaars die aan visuele elementen werken, tegelijkertijd kunnen samenwerken met ontwikkelaars die een omvattend SWF-bestand maken dat het SWF-bestand met grafische elementen bij uitvoering laadt. U kunt deze methode gebruiken om een serie bestandsversies te onderhouden, waarin grafische elementen niet afhankelijk zijn van de voortgang in de programmeringsontwikkeling.

De toepassing RuntimeAssetsExplorer laadt een SWF-bestand dat een subklasse van RuntimeAsset is en zorgt ervoor dat u door de beschikbare elementen van dat SWF-bestand kunt bladeren. Het voorbeeld illustreert het volgende:

- Een extern SWF-bestand laden met behulp van `Loader.load()`
- Het dynamisch maken van een bibliotheeksymbool dat voor ActionScript is geëxporteerd

- ActionScript-beheer voor het afspelen van filmclips

Zorg voordat u begint dat elk SWF-bestand dat in Flash Player moet worden uitgevoerd, zich in dezelfde beveiligingssandbox bevindt. Zie "Beveiligingssandboxen" op pagina 1108 voor meer informatie.

Als u wilt beschikken over de toepassingsbestanden voor dit voorbeeld, downloadt u de Flash Professional-voorbeeldbestanden. De toepassingsbestanden van RuntimeAssetsExplorer vindt u in de map Samples/RuntimeAssetsExplorer. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
RuntimeAssetsExample.mxml of RuntimeAssetsExample.fla	De gebruikersinterface voor de toepassing voor Flex (MXML) of Flash (FLA).
RuntimeAssetsExample.as	Klasse Document voor de Flash-toepassing (FLA).
GeometricAssets.as	Een voorbeeldklasse die de RuntimeAsset-interface implementeert.
GeometricAssets.fla	Een FLA-bestand dat is gekoppeld aan de klasse GeometricAssets (de documentklasse van de FLA), waarin symbolen zijn opgenomen die voor ActionScript zijn geëxporteerd.
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	Een interface waarin wordt gedefinieerd welke vereiste methoden worden verwacht van alle SWF-bestanden met uitvoeringselementen die in de verkennercontainer worden geladen.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	De klasse van het bibliotheeksymbool dat de vorm van een draaiend vak heeft.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	De klasse van het bibliotheeksymbool dat de vorm van een draaiende ster heeft.

Een uitvoeringsbibliotheekinterface tot stand brengen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u wilt zorgen dat de browser goed met een SWF-bibliotheek kan communiceren, moet de structuur van de bibliotheken met uitvoeringselementen worden geformaliseerd. Hiervoor wordt een interface gemaakt, die lijkt op een klasse (het is een blauwdruk van methoden die een verwachte structuur kunnen omsluiten), waarin zich echter geen methoden bevinden. De interface biedt een manier om de uitvoeringsbibliotheek en de browser met elkaar te laten communiceren. Elke SWF met uitvoeringselementen die in uw browser wordt geladen, implementeert deze interface. Zie het onderwerp over interfaces in *ActionScript 3.0 leren gebruiken* voor meer informatie over interfaces en hoe u deze kunt gebruiken.

De interface RuntimeLibrary is erg eenvoudig: er is slechts een functie voor nodig die de browser een array klassenpaden voor de te exporteren symbolen biedt en die beschikbaar is in de uitvoeringsbibliotheek. Hiervoor kent de interface een enkele methode: `getAssets()`.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

Het SWF-bestand met de elementenbibliotheek maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de interface RuntimeLibrary definieert, is het mogelijk meerdere SWF-bestanden met een elementenbibliotheek te maken die in een ander SWF-bestand kunnen worden geladen. Bij het maken van een individueel SWF-bestand met een elementenbibliotheek zijn vier taken betrokken:

- Een klasse voor het SWF-bestand met de elementenbibliotheek maken
- Klassen maken voor individuele elementen die zich in de bibliotheek bevinden
- De feitelijke grafische elementen maken
- Grafische elementen aan klassen koppelen en de bibliotheek-SWF publiceren

Een klasse maken om de interface RuntimeLibrary te implementeren

We maken nu de klasse GeometricAssets, die de interface RuntimeLibrary implementeert. Dit wordt de documentklasse van de FLA. De code voor deze klasse lijkt erg op de interface RuntimeLibrary: het verschil is dat de methode `getAssets()` in de klassendefinitie een methode heeft.

```
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" ];
        }
    }
}
```

Als we bijvoorbeeld een tweede uitvoeringsbibliotheek maken, kunnen we een andere FLA maken, die is gebaseerd op een andere klasse (bijvoorbeeld AnimationAssets) met een eigen implementatie van `getAssets()`.

Klassen voor elk MovieClip-element maken

In dit voorbeeld breiden we de klasse MovieClip slechts uit, zonder dat functionaliteit wordt toegevoegd aan de aangepaste elementen. De volgende code voor AnimatingStar is gelijk aan de code van AnimatingBox:

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

De bibliotheek publiceren

We koppelen de op MovieClip gebaseerde elementen nu aan de nieuwe klasse door een nieuw FLA-bestand te maken en GeometricAssets in het veld Documentklasse van Eigenschapcontrole in te voeren. In dit voorbeeld maken we twee eenvoudige vormen, die gebruik maken van een tijdlijn-tween voor een rotatie over 360 frames met de klok mee. Zowel de symbolen `animatingBox` als `animatingStar` zijn ingesteld op Exporteren voor ActionScript. Het veld Klasse is ingesteld op de respectievelijke klassenpaden, die in de implementatie `getAssets()` zijn opgegeven. De standaardbasisklasse `flash.display.MovieClip` blijft behouden, omdat we een subklasse van de standaard MovieClip-methoden willen maken.

Na het maken van de exportinstellingen voor uw symbool, publiceert u de FLA. U hebt nu uw eerste uitvoeringsbibliotheek gemaakt. Deze SWF kan in een ander AVM2 SWF-bestand worden geladen, zodat de symbolen AnimatingBox en AnimatingStar in het nieuwe SFW-bestand beschikbaar zijn.

De bibliotheek in een ander SWF-bestand laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De laatste functie die moet worden uitgevoerd, is de gebruikersinterface voor de elementverkenner. In dit voorbeeld heeft het pad naar de uitvoeringsbibliotheek een vaste structuur, met de naam `ASSETS_PATH` als variabele. U kunt ook de klasse FileReference gebruiken om bijvoorbeeld een interface te maken, die op de vaste schijf naar een bepaald SWF-bestand bladert.

Wanneer het laden van de uitvoeringsbibliotheek is voltooid, roept Flash Player de methode `runtimeAssetsLoadComplete()` aan:

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

In deze methode vertegenwoordigt de variabele rl het geladen SWF-bestand. De code roept de methode `getAssets()` van het geladen SWF-bestand aan, waarbij de lijst wordt verkregen met elementen die beschikbaar zijn. De elementen worden vervolgens gebruikt om een component ComboBox te vullen met een lijst beschikbare elementen door de methode `populateDropDown()` aan te roepen. Deze methode slaat op zijn beurt het volledige klassenpad van elk element op. Wanneer u in de gebruikersinterface op de knop Toevoegen klikt, wordt de methode `addAsset()` geactiveerd:

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

het klassenpad van het op dat moment geselecteerde element in ComboBox (`assetNameCbo.selectedItem.data`) wordt opgehaald en de functie `getDefinitionByName()` (uit het pakket flash.utils) wordt gebruikt om een daadwerkelijke verwijzing naar de klasse van het element te verkrijgen, zodat een nieuwe instantie van dat element kan worden gemaakt.

Hoofdstuk 17: Werken met bewegingstweens

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

In “Animaties van objecten” op pagina 204 wordt beschreven hoe u animatiescripts in ActionScript implementeert.

Hier beschrijven we een andere techniek voor het maken van animatie: bewegingstweens. Met deze techniek kunt u beweging creëren door de beweging met Adobe® Flash® Professional interactief in een document in te stellen. Vervolgens kunt u deze beweging tijdens de runtime gebruiken in een dynamische, op ActionScript gebaseerde animatie.

Flash Professional genereert automatisch de ActionScript die de bewegingstween implementeert en deze beschikbaar maakt, zodat u deze kunt kopiëren en hergebruiken.

U moet een licentie voor Flash Professional hebben om bewegingstweens te maken.

Meer Help-onderwerpen

fl.motion-pakket

Basisinformatie over bewegingstweens

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

Bewegingstweens vormen een eenvoudige manier om animatie te maken.

Een bewegingstween wijzigt per frame de eigenschappen van een weergaveobject, zoals positie of rotatie. Door verschillende filters en andere eigenschappen toe te passen, kan een bewegingstween ook de weergave van een weergaveobject wijzigen terwijl het beweegt. U kunt de bewegingstween interactief creëren met behulp van Flash Professional, die de ActionScript voor de bewegingstween genereert. Vanuit Flash gebruikt u de opdracht Beweging kopiëren als ActionScript 3.0 om de ActionScript-code te kopiëren die de bewegingstween heeft gemaakt. Daarna kunt u de ActionScript-code hergebruiken om tijdens de runtime beweging te maken in uw eigen dynamische animatie.

Zie de sectie Bewegingstweens in *Flash Professional gebruiken* voor informatie over het maken van een bewegingstween.

Belangrijke concepten en termen

Hier volgt een belangrijke term die u in dit hoofdstuk zult tegenkomen:

Bewegingstween Een constructie die tussenliggende frames van een weergaveobject in verschillende toestanden op verschillende momenten genereert; schept de indruk dat de eerste toestand geleidelijk voortvloeit in de tweede. Wordt gebruikt om een weergaveobject over het werkgebied voort te bewegen, of om het na verloop van tijd te doen groeien, krimpen, draaien, vervagen of van kleur te veranderen.

Scripts voor bewegingstweens kopiëren in Flash

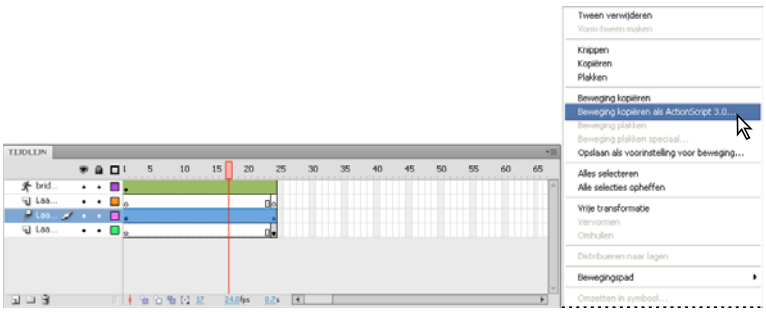
Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

Een tween genereert tussenliggende frames die een weergaveobject in verschillende toestanden weergeven in twee verschillende frames op een tijdlijn. Zo wordt de indruk gewekt dat de afbeelding in het eerste frame geleidelijk overgaat in de afbeelding in het tweede frame. In een bewegingstween heeft de wijziging in de verschijning doorgaans tot gevolg dat de positie van het weergaveobject verandert, waardoor beweging ontstaat. Behalve het verplaatsen van het weergaveobject kan een bewegingstween een object ook draaien, scheeftrekken, van formaat veranderen of er filters op toepassen.

Maak een bewegingstween in Flash door een weergaveobject tussen hoofdframes langs de tijdlijn te verplaatsen. Flash genereert automatisch de ActionScript-code die de tween beschrijft. U kunt deze code kopiëren en in een bestand opslaan. Zie de sectie Bewegingstweens in *Flash Professional gebruiken* voor informatie over het maken van een bewegingstween.

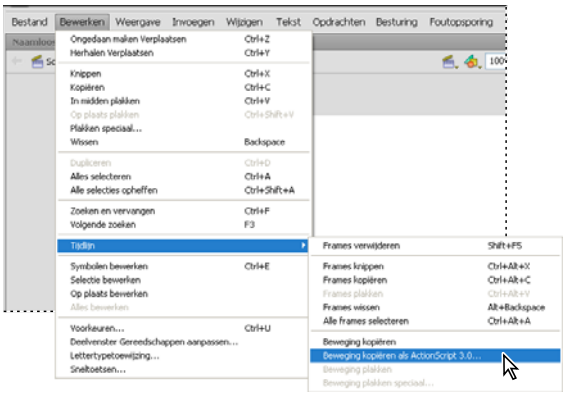
U kunt de opdracht Beweging kopiëren als ActionScript 3.0 in Flash op twee manieren benaderen. Ten eerste, vanuit het snelmenu van een tween in het werkgebied:

1. Selecteer de bewegingstween in het werkgebied.
2. Klik met de rechtermuisknop (Windows) of houd Control ingedrukt terwijl u klikt (Macintosh).
3. Kies Beweging kopiëren als ActionScript 3.0 . . .



De tweede manier is door de opdracht rechtstreeks te kiezen in het menu Bewerken van Flash:

1. Selecteer de bewegingstween in het werkgebied.
2. Selecteer Bewerken > Tijdlijn > Beweging kopiëren als ActionScript 3.0.



Nadat u het script hebt gekopieerd, plakt u het in een bestand en slaat u het bestand op.

Nadat een bewegingstween is gemaakt en het script is gekopieerd en opgeslagen, kunt u het ongewijzigd hergebruiken, maar u kunt het ook wijzigen in uw eigen dynamische, op ActionScript gebaseerde animatie.

Scripts voor bewegingstweens opnemen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De header in de ActionScript-code die u uit Flash kopieert, vermeldt alle modules die nodig zijn ter ondersteuning van de bewegingstween.

Bewegingstweenklassen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De belangrijkste klassen zijn AnimatorFactory, MotionBase en Motion uit het pakket `fl.motion`. Afhankelijk van de eigenschappen die de bewegingstween manipuleert, hebt u mogelijk aanvullende klassen nodig. Als de bewegingstween bijvoorbeeld het weergaveobject transformeert of draait, importeert u de desbetreffende `flash.geom`-klassen. Als de tween filters toepast, importeert u de `flash.filter`-klassen. In ActionScript is een bewegingstween een instantie van de klasse Motion. De klasse Motion slaat een hoofdframe-animatiereeks op die op een visueel object kan worden toegepast. De animatiegegevens hebben betrekking op positie, schaal, rotatie, scheeftrekking, kleur, filters en versnelling.

De volgende ActionScript-code is gekopieerd uit een bewegingstween die is gemaakt om een weergaveobject met de instantienaam `Symbol1_2` te animeren. In de code wordt een variabele gedeclareerd voor het MotionBase-object `__motion_Symbol1_2`. De klasse MotionBase is de bovenliggende klasse van de klasse Motion.

```
var __motion_Symbol1_2:MotionBase;
```

Daarna wordt in het script het Motion-object gemaakt:

```
__motion_Symbol1_2 = new Motion();
```

Motion-objectnamen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

In het vorige geval genereert Flash automatisch de naam `__motion_Symbol1_2` voor het Motion-object. Het voorvoegsel `__motion_` wordt aan de naam van het weergaveobject toegevoegd. De automatisch gegenereerde naam is dus gebaseerd op de instantienaam van het doelobject van de bewegingstween in het Flash-programma voor het schrijven van programmacode. De eigenschap `duration` van het Motion-object geeft het totale aantal frames in de bewegingstween aan:

```
__motion_Symbol1_2.duration = 200;
```

Standaard krijgt de instantie van het weergaveobject waarvan u de bewegingstween wilt kopiëren automatisch een naam als deze nog geen naam heeft.

Wanneer u ActionScript die door Flash is gemaakt opnieuw gebruikt in uw eigen animatie, kunt u de naam houden die Flash automatisch voor de tween genereert of kunt u deze vervangen met een andere naam. Als u de naam van de tween verandert, moet u erop letten dat u deze in het hele script aanpast.

Een andere mogelijkheid is dat u in Flash een zelfgekozen naam toekent aan het doelobject van de bewegingstween. Maak vervolgens de bewegingstween en kopieer het script. Welke benadering u ook gebruikt, let erop dat elk Motion-object in uw ActionScript-code een unieke naam heeft.

Animaties beschrijven

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De methode `addPropertyArray()` van de klasse MotionBase voegt een array van waarden toe om elke eigenschap met tweening te beschrijven.

Mogelijk bevat de array één arrayitem voor elk hoofdframe in de bewegingstween. Dikwijls bevatten sommige arrays minder items dan het totale aantal hoofdframes in de bewegingstween. Deze situatie doet zich voor wanneer de laatste waarde in de array niet verandert voor de overige frames.

Als de lengte van het arrayargument groter is dan de eigenschap `duration` van het Motion-object, past `addPropertyArray()` de waarde van de eigenschap `duration` dienovereenkomstig aan. Deze methode voegt geen hoofdframes toe voor de eigenschappen die eerder zijn toegevoegd. De zojuist toegevoegde hoofdframes blijven aanwezig in de extra frames van de animatie.

De eigenschappen `x` en `y` van het Motion-object beschrijven de veranderende positie van het getweende object terwijl de animatie wordt uitgevoerd. Deze coördinaten zijn de waarden die het meest waarschijnlijk in elk hoofdframe veranderen als de positie van het weergaveobject verandert. U kunt extra bewegingseigenschappen toevoegen met de methode `addPropertyArray()`. Voeg bijvoorbeeld de waarden `scaleX` en `scaleY` toe als het formaat van het getweende object wordt gewijzigd. Voeg de waarden `skewX` en `skewY` toe als het object wordt scheefgetrokken. Voeg de eigenschap `rotationConcat` toe als het object wordt geroteerd.

Gebruik de methode `addPropertyArray()` om de volgende tweeneigenschappen te definiëren:

`x`	horizontale positie van het transformatiepunt van het object in de coördinaatruimte van het bovenliggende object
`y`	verticale positie van het transformatiepunt van het object in de coördinaatruimte van het bovenliggende object
`z`	diepteposities (z-as) van het transformatiepunt van het object in de coördinaatruimte van het bovenliggende object
`scaleX`	horizontale schaal als een percentage van het object aangegeven zoals deze vanaf het transformatiepunt wordt toegepast
`scaleY`	verticale schaal als een percentage van het object aangegeven zoals deze vanaf het transformatiepunt wordt toegepast
`skewX`	horizontale scheeftrekhoek van het object in graden aangegeven zoals deze vanaf het transformatiepunt wordt toegepast
`skewY`	verticale scheeftrekhoek van het object in graden aangegeven zoals deze vanaf het transformatiepunt wordt toegepast
`rotationX`	rotatie van het object rond de X-as ten opzichte van de oorspronkelijke oriëntatie
`rotationY`	rotatie van het object rond de Y-as ten opzichte van de oorspronkelijke oriëntatie
`rotationConcat`	rotatiewaarden (z-as) van het object in de beweging ten opzichte van de vorige oriëntatie die is toegepast vanaf het transformatiepunt
`useRotationConcat`	Als deze waarde is ingesteld, wordt het doelobject gedraaid wanneer `addPropertyArray()` gegevens voor beweging levert

`blendMode`	De waarde van de klasse BlendMode geeft de kleurencombinatie van het object op met afbeeldingen eronder
`matrix3D`	De eigenschap matrix3D als deze voor het hoofdframe is ingesteld; wordt gebruikt voor 3D-tweens; als deze wordt gebruikt, worden alle vorige transformatie-eigenschappen genegeerd
`rotationZ`	Z-asrotatie van het object, in graden, vanuit de oorspronkelijke oriëntatie ten opzichte van de bovenliggende 3D-container; wordt in plaats van rotationConcat voor 3D-tweens gebruikt

Welke eigenschappen in het automatisch gegenereerde script worden toegevoegd, is afhankelijk van de eigenschappen die in Flash aan de bewegingstween zijn toegewezen. U kunt bepaalde eigenschappen toevoegen, verwijderen of wijzigen wanneer u uw eigen versie van het script aanpast.

De volgende code wijst waarden toe aan de eigenschappen van de bewegingstween `__motion_Wheel`. In dit geval verandert de locatie van het getweende weergaveobject niet, maar draait het object ter plekke rond door de 29 frames van de bewegingstween te doorlopen. De waarden die aan de array `rotationConcat` zijn toegewezen, bepalen de rotatie. De andere eigenschapwaarden van deze bewegingstween veranderen niet.

```
__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"]);
```

In het volgende voorbeeld verplaatst het weergaveobject met de naam `Leaf_1` zich over het werkgebied. De arrays met de eigenschappen `x` en `y` bevatten verschillende waarden voor elk van de 100 frames in de animatie. Daarnaast draait het object om zijn `z`-as terwijl het over het werkgebied wordt verplaatst. De items in de array met `rotationZ`-eigenschappen bepalen de rotatie.

```
__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"]);
```

Filters toevoegen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

Als het doelobject van een bewegingstween filters bevat, worden deze filters toegevoegd met de methoden `initFilters()` en `addFilterPropertyArray()` van de klasse Motion.

De array met filters initialiseren

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De methode `initFilters()` initialiseert de filters. Het eerste argument van deze methode is een array met de volledig gekwalificeerde klassennamen van alle filters die op het weergaveobject zijn toegepast. Deze array met filternamen wordt gegenereerd vanuit de lijst met filters voor de bewegingstween in Flash. In uw exemplaar van het script kunt u de filters in het pakket `flash.filters` verwijderen uit of toevoegen aan deze array. De volgende oproep initialiseert de lijst met filters voor het doelweergaveobject. Deze past `DropShadowFilter`, `GlowFilter` en `BevelFilter` toe en kopieert de lijst naar elk hoofdframe in het Motion-object.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

Filters toevoegen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De methode `addFilterPropertyArray()` beschrijft de eigenschappen van een geïnitialiseerd filter met de volgende argumenten:

1. Het eerste argument identificeert een filter middels de index. De index verwijst naar de positie van de filternaam in de array met filterklassennamen die in een vorige oproep van `initFilters()` is doorgegeven.
2. Het tweede argument is de eigenschap filter die voor dat filter in elk hoofdframe moet worden opgeslagen.
3. Het derde argument is de waarde van de opgegeven eigenschap filter.

Op basis van de vorige oproep van `initFilters()`, wijzen de volgende oproepen van `addFilterPropertyArray()` de waarde 5 toe aan de eigenschappen `blurX` en `blurY` van `DropShadowFilter`. `DropShadowFilter` is het eerste item (index 0) in de geïnitialiseerde array met filters:

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

Met de volgende drie oproepen worden waarden toegewezen aan de eigenschappen quality, alpha en color van `GlowFilter`,, het tweede item (index 1) in de geïnitialiseerde array met filters:

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

Met de volgende vier oproepen worden waarden toegewezen aan `shadowAlpha`, `shadowColor`, `highlightAlpha` en `highlightColor` van `BevelFilter`, het derde item (index 2) in de geïnitialiseerde array met filters:

```
__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]);
```

Kleuren aanpassen met ColorMatrixFilter

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

Nadat `ColorMatrixFilter` is geïnitialiseerd, kunt u de desbetreffende `AdjustColor`-eigenschappen instellen om de helderheid, het contrast, de verzadiging en de tint van het getweende weergaveobject aan te passen. De `AdjustColor`-filter wordt toegepast wanneer de bewegingstween in Flash wordt gemaakt; u kunt deze verfijnen in uw versie van ActionScript. In het volgende voorbeeld worden de tint en verzadiging van het weergaveobject tijdens de beweging getransformeerd.

```
__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);
```

Bewegingstweens aan de bijbehorende weergaveobjecten koppelen

Flash Player 9 en hoger, Adobe AIR 1.0 en hoger, vereist Flash CS3 of hoger

De laatste taak is het koppelen van de bewegingstween aan de weergaveobjecten die deze manipuleert.

De klasse AnimatorFactory beheert de koppeling tussen een bewegingstween en de doelweergaveobjecten. Het argument voor de AnimatorFactory-constructor is het Motion-object:

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

Gebruik de methode `addTarget()` van de klasse AnimatorFactory om het doelweergaveobject aan de juiste bewegingstween te koppelen. De ActionScript-code die uit Flash is gekopieerd, markeert de regel `addTarget()` als commentaar en geeft geen instantienaam op:

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

Geef in uw exemplaar het weergaveobject op dat u aan de bewegingstween wilt koppelen. In het volgende voorbeeld zijn de doelen `greenWheel` en `redWheel` opgegeven:

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

U kunt meerdere weergaveobjecten aan dezelfde bewegingstween koppelen met behulp van oproepen van `addTarget()`.

Hoofdstuk 18: Werken met IK

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

IK (Inverse kinematics, Omgekeerde bewegingen) is een prachtige techniek voor het creëren van natuurlijke beweging.

Met IK kunt u gecoördineerde bewegingen maken binnen een keten van verbonden onderdelen, een zogenaamde IK-armatuur, zodat de onderdelen samen bewegen als op een natuurlijke manier. De onderdelen van de armatuur worden de beenderen en gewrichten genoemd. Aan de hand van het eindpunt van de armatuur berekent IK de hoeken voor de gewrichten die nodig zijn om dat eindpunt te bereiken.

Het handmatig berekenen van deze hoeken zou een hele opgave zijn. Het mooie van deze functie is dat u interactief armaturen kunt maken met Adobe® Flash® Professional. Daarna kunt u ze laten bewegen met ActionScript. Het IK-systeem dat in Flash Professional voor het schrijven van programmacode is opgenomen, voert de berekeningen uit die de beweging van de armatuur beschrijven. U kunt de beweging beperken tot bepaalde parameters in uw ActionScript-code.

In de Flash Professional CS5-versie van IK wordt boonvering geïntroduceerd, een functie die meestal wordt aangetroffen in geavanceerde animatietoepassingen. In combinatie met de nieuwe, dynamische fysische engine kunt u zo levensechte bewegingen configureren. En het effect is zowel tijdens runtime zichtbaar als tijdens de ontwerpfase.

U moet een licentie voor Flash Professional hebben om IK-armaturen te maken.

Meer Help-onderwerpen

fl.ik-pakket

Basisinformatie over IK

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

Met IK kunt u natuurlijke bewegingen maken door onderdelen zo aan elkaar te koppelen dat ze op een realistische wijze ten opzichte van elkaar bewegen.

Met IK kunt u bijvoorbeeld één been naar een bepaalde positie verplaatsen door de bewegingen van de gewrichten in het been te construeren die nodig zijn om de gewenste houding te verkrijgen. IK gebruikt een framework van beenderen die aan elkaar zijn gekoppeld in een structuur die een IK-armatuur wordt genoemd. Het pakket `fl.ik` helpt u bij het maken van natuurlijke bewegingen. Hiermee kunt u meerdere IK-armaturen naadloos laten bewegen, zonder dat u veel kennis hoeft te hebben van de natuurkundige facetten achter de IK-algoritmen.

Maak de IK-armatuur met de hulpbeenderen en -gewrichten in Flash Professional. Daarna kunt u de IK-klassen benaderen om deze tijdens de runtime te animeren.

Raadpleeg de sectie IK gebruiken in *Flash Professional gebruiken* voor gedetailleerde instructies over het maken van een IK-armatuur.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen die relevant zijn voor deze functie:

Armatuur Een bewegingsketen, bestaande uit beenderen en gewrichten, die in een computeranimatie wordt gebruikt om realistische bewegingen te simuleren.

Bone Een star segment in een armatuur, te vergelijken met een bot in een skelet.

IK (Inverse kinematics ofwel omgekeerde bewegingen) Het bepalen van de parameters van een samengevoegd flexibel object, dat een bewegingsketen of armatuur wordt genoemd.

Gewricht De plaats waar twee beenderen elkaar raken, geconstrueerd om beweging van de beenderen mogelijk te maken; te vergelijken met een gewricht in een lichaam.

Fysische engine Een pakket met fysische algoritmen die worden gebruikt om de bewegingen van animaties levensecht te laten overkomen.

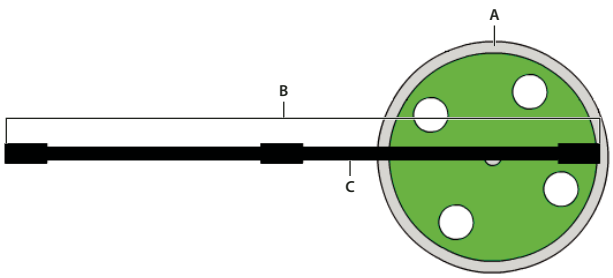
Veer De eigenschap dat een been, wanneer het bovenliggende been wordt bewogen, beweegt en reageert De beweging vermindert naar verloop van tijd.

Animaties van IK-armaturen - overzicht

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

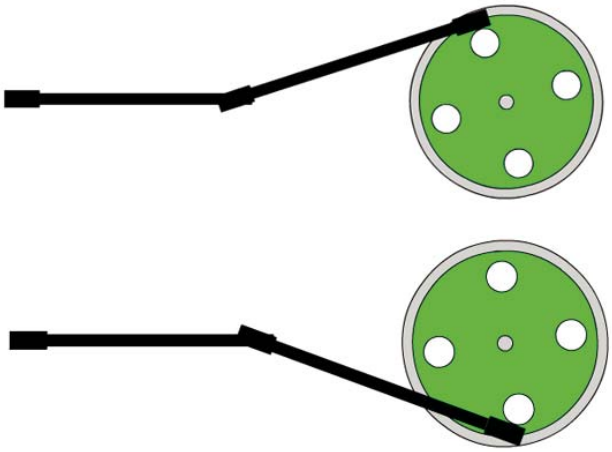
Gebruik na het maken van een IK-armatuur de `fl.ik`-klassen om de beweging te beperken, gebeurtenissen te volgen en de armatuur tijdens de runtime te animeren.

In de volgende afbeelding ziet u het filmfragment `Wheel`. De as is een instantie van een IKArmature met de naam `Axle`. De klasse IKMover beweegt de armatuur synchroon met de rotatie van het wiel. De IKBone, `ikBone2`, in de armatuur is via het eindgewricht aan het wiel gekoppeld.



***A.** Wiel **B.** As **C.** ikBone2*

Tijdens de runtime draait het wiel gekoppeld aan de bewegingstween `__motion_Wheel` die is beschreven in "Animaties beschrijven" op pagina 357. Een IKMover-object initieert en bestuurt de beweging van de as. In de volgende afbeelding ziet u twee momentopnamen van de asarmatuur, gekoppeld aan het draaiende wiel, in verschillende frames in de rotatie.



Tijdens de runtime zal het volgende ActionScript:

- informatie ophalen over de armatuur en de componenten daarvan;
- een IKMover-object instantiëren;
- de as samen met de rotatie van het wiel bewegen.

```
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);
    }
}
```

De volgende IK-klassen worden gebruikt om de as te bewegen:

- IKArmature: beschrijft de armatuur, een boomstructuur die uit beenderen en gewrichten bestaat; moet in Flash Professional zijn gemaakt
- IKManager: containerklasse voor alle IK-armaturen in het document; moet in Flash Professional zijn gemaakt.
- IKBone: een segment van een IK-armatuur
- IKJoint: een verbinding tussen twee IK-beenderen
- IKMover: initieert en bestuurt de IK-beweging van armaturen

Zie het IK-pakket voor een complete en uitgebreide beschrijving van deze klassen.

Informatie krijgen over een IK-armatuur

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

Eerst declareert u variabelen voor de armatuur, het been en het gewricht waaruit de onderdelen bestaan die u wilt laten bewegen.

In de volgende code wordt de methode `getArmatureByName()` van de klasse IKManager gebruikt om de waarde van de Axle-armatuur toe te kennen aan de IKArmature-variabele `tree`. De Axle-armatuur is eerder met Flash Professional gemaakt.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

Evenzo wordt in de volgende code de methode `getBoneByName()` van de klasse IKArmature gebruikt om aan de IKBone-variabele de waarde van het been `ikBone2` toe te kennen.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

Het eindgewricht van het been `ikBone2` is het onderdeel van de armatuur dat aan het draaiende wiel wordt vastgemaakt.

Op de volgende regel wordt de variabele `endEffector` gedeclareerd en krijgt deze de waarde van de eigenschap `tailjoint` van het been `ikBone2`:

```
var endEffector:IKJoint = home.tailjoint;
```

De variabele `pos` is een punt dat de huidige positie van het gewricht `endEffector` bevat.

```
var pos:Point = endEffector.position;
```

In dit voorbeeld is `pos` de positie van het gewricht aan het einde van de as, waar deze aan het wiel is gekoppeld. De oorspronkelijke waarde van deze variabele is verkregen uit de eigenschap `position` van de IKJoint.

IK-bewegingsobjecten instantiëren en de beweging beperken

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

Een instantie van de klasse IKMover laat de as bewegen.

Op de volgende regel wordt het IKMover-object `ik` geïnstantieerd en worden aan de constructor van dit object het te bewegen object en het beginpunt van de beweging doorgegeven:

```
var ik:IKMover = new IKMover(endEffector, pos);
```

Met de eigenschappen van de klasse IKMover kunt u de beweging van een armatuur beperken. U kunt een beweging beperken op basis van afstand, herhalingen en de tijd van de beweging.

Met de volgende eigenschappenparen worden deze beperkingen afgedwongen. De paren bestaan uit een Boolean die aangeeft of de beweging wordt beperkt, en een getal dat de beperking opgeeft:

Boolean-eigenschap	Integer-eigenschap	Limit-instelling
`limitByDistance:Boolean`	`distanceLimit:int`	Stelt de maximumafstand in pixels in die de IK-engine voor elke herhaling aflegt.
`limitByIteration:Boolean`	`iterationLimit:int`	Stelt het maximum aantal herhalingen in dat de IK-engine voor elke beweging uitvoert.
`limitByTime:Boolean`	`timeLimit:int`	Stelt de maximumtijd in milliseconden in die aan het IK-systeem is toegekend om de beweging uit te voeren.

Standaard zijn alle Booleans ingesteld op `false`. Dat betekent dat de beweging niet wordt beperkt, tenzij u dit expliciet `true`opgeeft. Om een beperking wilt afdwingen, stelt u de gerelateerde eigenschap in op `true` en geeft u een waarde op voor de gerelateerde integer-eigenschap. Als u de beperking op een waarde instelt zonder de corresponderende Boolean in te stellen, wordt de beperking genegeerd. In dat geval blijft de IK-engine het object bewegen tot een andere beperking of de doelpositie van de IKMover is bereikt.

In het volgende voorbeeld is de maximumafstand van de armatuurbeweging ingesteld op 0,1 pixel per herhaling. Het maximum aantal herhalingen voor elke beweging is ingesteld op tien.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

IK-armaturen verplaatsen

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

De IKMover beweegt de as binnen de gebeurtenislistener voor het wiel. Bij elke enterFrame-gebeurtenis van het wiel wordt een nieuwe doelpositie voor de armatuur berekend. Met de methode `moveTo()` beweegt de IKMover het eindgewricht naar de doelpositie of zover mogelijk binnen de beperkingen die zijn ingesteld met de eigenschappen `limitByDistance`, `limitByIteration` en `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);
    }
}
```

Vering gebruiken

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS5 of hoger

IK biedt in Flash Professional CS5 bovendien ondersteuning voor beenvering. Beenvering kan tijdens de ontwerpfase worden ingesteld en kenmerken voor beenvering kunnen tijdens runtime worden toegevoegd of gewijzigd. De vering is een eigenschap van een been en de gewrichten van dit been. De vering heeft twee kenmerken, namelijk `IKJoint.springStrength` en `IKJoint.springDamping`. Door het eerste kenmerk wordt de hoeveelheid vering ingesteld en door het tweede kenmerk wordt weerstand aan de hoeveelheid vering toegevoegd. Het tweede kenmerk wijzigt eveneens de snelheid waarmee de vering vermindert.

De hoeveelheid vering is een procentuele waarde, variërend van de standaardinstelling 0 (totaal onbuigzaam) tot 100 (bijzonder flexibel en bestuurd door fysica). Benen met vering reageren op de beweging van het bijbehorende gewricht. Als er geen andere translatie is ingesteld (rotatie, x of y), hebben de veringsinstellingen geen effect.

De demping is een procentuele waarde, variërend van de standaardinstelling 0 (geen weerstand) tot 100 (bijzonder sterke weerstand). De demping verandert de hoeveelheid tijd tussen de eerste beweging van een been en het moment waarop het been weer terugkeert in de ruststand.

U kunt zien of veringen geschikt zijn voor een IKArmature-object door de eigenschap `IKArmature.springsEnabled` van het object te controleren. De andere veringseigenschappen en -methoden horen bij afzonderlijke IKJoint-objecten. Een gewricht kan geschikt worden gemaakt voor hoekrotatie en translatie langs de x- en y-assen. U kunt de veringshoek van een roterend gewricht positioneren met `IKJoint.setSpringAngle` en de veringspositie van een translerend gewricht met `IKJoint.setSpringAngle`.

In dit voorbeeld wordt een been op naam geselecteerd en wordt de tailJoint van het been geïdentificeerd. De code controleert de bovenliggende armatuur om te zien of veringen zijn ingeschakeld en stelt vervolgens veringseigenschappen voor het gewricht in.

```
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
    }
}
```

IK-gebeurtenissen gebruiken

Flash Player 10 en hoger, Adobe AIR 1.5 en hoger, vereist Flash CS4 of hoger

Met de klasse IKEvent kunt u een gebeurtenisobject maken dat informatie over IK-gebeurtenissen bevat. IKEvent-gegevens beschrijven beweging die is beëindigd omdat de opgegeven beperking voor tijd, afstand of herhaling is overschreden.

De volgende code illustreert een gebeurtenislistener en -handler voor het bijhouden van tijdbeperkingsgebeurtenissen. Deze gebeurtenishandler brengt rapport uit over de tijd, de afstand, het aantal herhalingen en de gewrichten van een gebeurtenis die wordt geactiveerd wanneer de tijdbeperking voor de IKMover is overschreden.

```
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);
}
```

Hoofdstuk 19: Werken in drie dimensies (3D)

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De Flash Player- en AIR-runtimes bieden op twee manieren ondersteuning voor 3D-afbeeldingen. U kunt driedimensionale weergaveobjecten in de Flash-weergavelijst gebruiken. Dat is geschikt voor het toevoegen van 3D-effecten aan Flash-inhoud en voor objecten met weinig veelhoeken. In Flash Player 11 en AIR 3 of later kunt u complexe 3D-scènes renderen met de Stage3D API.

Een Stage3D-viewport is geen weergaveobject. De 3D-afbeeldingen worden in plaats daarvan gerenderd naar een viewport die onder de Flash-weergavelijst (en boven de StageVideo-viewportvlakken) wordt weergegeven. In plaats van de klassen Flash DisplayObject te gebruiken om de scène te maken, gebruikt u een programmeerbare 3D-pipeline (vergelijkbaar met OpenGL en Direct3D). Deze pipeline gebruikt driehoeksgegevens en -structuren als invoer en rendert de scène met gebruik van de shaderprogramma's die u verschaft. Hardwareversnelling wordt gebruikt wanneer op de clientcomputer een compatibele GPU met ondersteunde stuurprogramma's beschikbaar is.

Stage3D verschaft een API van bijzonder laag niveau. In een toepassing wordt u aangeraden een 3D-framework te gebruiken dat ondersteuning biedt voor Stage3D. U kunt natuurlijk uw eigen framework gebruiken, maar u kunt ook een van de verschillende open-sourceframeworks gebruiken die al in de handel verkrijgbaar zijn.

Ga naar Flash Player Developer Center: Stage 3D voor meer informatie over het ontwikkelen van 3D-toepassingen met gebruik van Stage3D en over beschikbare 3D-frameworks.

Basisbeginselen van 3D-weergaveobjecten

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het belangrijkste verschil tussen een tweedimensionaal (2D) en een driedimensionaal (3D) object dat wordt geprojecteerd op een tweedimensionaal scherm, is de toevoeging van een derde dimensie aan het object. De derde dimensie maakt het mogelijk het object te bewegen naar het gezichtspunt van de gebruiker en er vandaan.

Wanneer u de eigenschap `z` van een weergaveobject expliciet instelt op een numerieke waarde, maakt het object automatisch een 3D-transformatiematrix. U kunt deze matrix aanpassen om de 3D-transformatie-instellingen van dat object te wijzigen.

Bovendien is 3D-rotatie anders dan 2D-rotatie. In 2D staat de rotatieas altijd loodrecht op het x/y-vlak, oftewel op de z-as. In 3D kan om elke as worden gedraaid, zowel om de x- als de y- of z-as. Door de rotatie- en schalingseigenschappen van een weergaveobject in te stellen, kunt u het verplaatsen in de 3D-ruimte.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het programmeren van 3D-afbeeldingen.

Perspectief In een 2D-vlak is dit de representatie van parallelle lijnen alsof deze elkaar in een verdwijnpunt raken om de indruk van diepte en afstand te wekken.

Projectie De productie van een 2D-afbeelding van een object met meer dimensies; 3D-projectie wijst 3D-punten toe aan een 2D-vlak.

Rotatie De richting (en vaak de positie) van een object wijzigen door elk punt dat deel uitmaakt van het object, te bewegen in een ronde beweging.

Transformatie 3D-punten of puntensets wijzigen door transleren, roteren, schalen, scheeftrekken of een combinatie van deze acties.

Transleren De positie van een object wijzigen door elk punt dat deel uitmaakt van het object, met dezelfde waarde in dezelfde richting te verplaatsen.

Verdwijnpunt Punt waarop van de kijker af lopende parallelle lijnen elkaar lijken te raken wanneer deze worden gerepresenteerd in lineair perspectief.

Vector Een 3D-vector stelt een punt of een locatie in de driedimensionale ruimte voor met behulp van de cartesische coördinaten x, y en z.

Vertex Een hoekpunt.

Structuurnet Elk punt dat een object in de 3D-ruimte definieert.

UV-toewijzing Een manier om een structuur of bitmap toe te wijzen aan een 3D-oppervlak. Bij UV-toewijzing worden waarden aan coördinaten op een afbeelding toegewezen als percentage van de horizontale (U) en verticale (V) as.

T-waarde De schaalfactor voor het bepalen van de grootte van een 3D-object als het object wordt verplaatst in de richting van het huidige gezichtspunt of er vandaan.

Schifting Renderen of niet renderen van oppervlakken met een specifieke wenteling. Met behulp van schifting kunt u oppervlakken verbergen die vanuit het huidige gezichtspunt niet zichtbaar zijn.

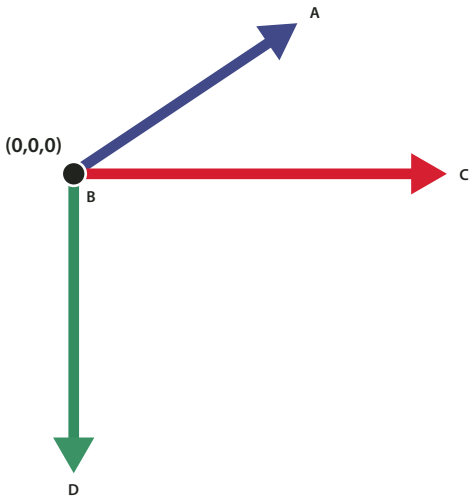
Inzicht in 3D-weergaveobjecten in Flash Player en de AIR-runtime

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In Flash Player-versies vóór Flash Player 10 en in Adobe AIR-versies vóór Adobe AIR 1.5 hebben weergaveobjecten twee eigenschappen, `x` en `y`, om ze op een 2D-vlak te plaatsen. Vanaf Flash Player 10 en Adobe AIR 1.5 heeft elk ActionScript-weergaveobject de eigenschap `z`, waarmee u het object langs de z-as kunt plaatsen die doorgaans wordt gebruikt om diepte of afstand aan te geven.

Flash Player 10 en Adobe AIR 1.5 bevat ondersteuning voor 3D-effecten. Weergaveobjecten zelf zijn echter plat. Elk weergaveobject, zoals een MovieClip- of Sprite-object, wordt gerenderd in twee dimensies, op één vlak. Met de 3D-functies kunt u deze vlakke objecten in alle drie dimensies plaatsen, verplaatsen, roteren en anderszins transformeren. U kunt ook 3D-punten beheren en omzetten naar 2D x- en y-coördinaten zodat u 3D-objecten kunt projecteren op een 2D-weergave. U kunt een groot aantal 3D-ervaringen simuleren met deze functies.

Het 3D-coördinatenstelsel dat ActionScript gebruikt, wijkt af van andere stelsels. Wanneer u in ActionScript 2D-coördinaten gebruikt, wordt de waarde van x hoger als u langs de x-as naar rechts beweegt en wordt de waarde van y hoger als u langs de y-as omlaag beweegt. Het 3D-coördinatenstelsel houdt zich aan deze conventies en voegt hier een z-as aan toe waarvan de waarden hoger worden als u van het gezichtspunt af beweegt.



De positieve richtingen van de x-, y- en z-as in het 3D-coördinatenstelsel van ActionScript.
***A.** +Z-as **B.** Oorsprong **C.** +X-as **D.** +Y-as*

***Opmerking:** Houd er rekening mee dat Flash Player en AIR 3D altijd in lagen representeert. Dit betekent dat als object A in de weergavelijst vóór object B komt, Flash Player of AIR A altijd vóór B rendert, ongeacht de waarde van de z-as van de twee objecten. U lost dit conflict tussen de volgorde in de weergavelijst en de volgorde op de x-as op door de lagen van 3D-weergaveobjecten op te slaan met de methode* `transform.getRelativeMatrix3D()` *en vervolgens een andere volgorde te geven. Zie "Matrix3D-objecten gebruiken om de weergavevolgorde te wijzigen" op pagina 382 voor meer informatie.*

De volgende ActionScript-klassen ondersteunen de nieuwe 3D-functies:

1. De klasse flash.display.DisplayObject beschikt over de eigenschap `z` en nieuwe rotatie- en schalingseigenschappen voor het bewerken van weergaveobjecten in de 3D-ruimte. De methode `DisplayObject.local3DToGlobal()` biedt een eenvoudige manier om 3D-geometrie te projecteren op een 2D-vlak.
2. De klasse flash.geom.Vector3D kan worden gebruikt als gegevensstructuur voor het beheer van 3D-punten. De klasse ondersteunt ook rekenkundige methoden voor vectoren.
3. De klasse flash.geom.Matrix3D ondersteunt complexe transformaties van 3D-geometrie, zoals roteren, schalen en transleren.
4. De klasse flash.geom.PerspectiveProjection bestuurt de parameters voor het toewijzen van 3D-geometrie aan een 2D-weergave.

Er zijn twee verschillende manieren om 3D-afbeeldingen te simuleren in ActionScript:

1. Vlakke objecten rangschikken en animeren in de 3D-ruimte. Bij deze aanpak voegt u animatie aan weergaveobjecten toe met behulp van de eigenschappen x, y en z van weergaveobjecten, of stelt u rotatie- en schalingseigenschappen in met behulp van de klasse DisplayObject. Complexere bewegingen kunt u bereiken met behulp van het DisplayObject.transform.matrix3D-object. Het DisplayObject.transform.perspectiveProjection-object past aan hoe de weergaveobjecten worden getekend in 3D-perspectief. Gebruik deze aanpak als u animatie wilt toevoegen aan 3D-objecten die hoofdzakelijk uit vlakken bestaan. Voorbeelden van deze aanpak zijn 3D-afbeeldingengalerieën of 2D-animatieobjecten die in de 3D-ruimte zijn gerangschikt.

2 2D-driehoeken genereren op basis van 3D-geometrie en deze driehoeken renderen met structuren. Als u deze aanpak kiest, moet u eerst gegevens over 3D-objecten definiëren en beheren, en deze gegevens vervolgens converteren naar 2D-driehoeken die kunnen worden gerenderd. U kunt bitmapstructuren toewijzen aan deze driehoeken. De driehoeken worden vervolgens naar een grafisch object getekend met de methode `Graphics.drawTriangles()`. Voorbeelden van deze aanpak zijn het laden van 3D-modelgegevens uit een bestand en het model op het scherm renderen, of 3D-terrein genereren en tekenen als driehoekige netten.

3D-weergaveobjecten maken en verplaatsen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een 2D-weergaveobject converteren naar een 3D-weergaveobject door de eigenschap `z` expliciet in te stellen op een numerieke waarde. Wanneer u een waarde toewijst aan de eigenschap `z`, wordt een nieuw Transform-object voor het weergaveobject gemaakt. Door de eigenschap `DisplayObject.rotationX` of `DisplayObject.rotationY` in te stellen, wordt ook een nieuw Transform-object gemaakt. Het Transform-object heeft de eigenschap `Matrix3D` die regelt hoe het weergaveobject wordt gerepresenteerd in de 3D-ruimte.

De volgende code stelt de coördinaten in voor een weergaveobject met de naam “leaf”:

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

U kunt deze waarden, en de eigenschappen die zijn afgeleid van deze waarden, zien in de eigenschap `matrix3D` van het Transform-object van "leaf":

```
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);
```

Zie de klasse Transform voor meer informatie over de eigenschappen van het Transform-object. Zie de klasse Matrix3D voor meer informatie over de eigenschappen van het Matrix3D-object.

Objecten verplaatsen in een 3D-ruimte

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een object verplaatsen in de 3D-ruimte door de waarden van de eigenschap `x, y of z` te wijzigen. Door de waarde van de eigenschap `z` te wijzigen, lijkt het object dichterbij of verder weg van de kijker te worden verplaatst.

De volgende code verplaatst twee ellipsen naar achteren en naar voren langs hun x-as door de waarde van de eigenschap `z` te wijzigen als reactie op een gebeurtenis. `ellipse2` beweegt sneller dan `ellipse1`: de eigenschap `z` van ellipse2 wordt bij elke framegebeurtenis verhoogd met een veelvoud van 20, terwijl de eigenschap `z` van `ellipse1` wordt verhoogd met een veelvoud van 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;
    }
}
```

Objecten roteren in de 3D-ruimte

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een object op drie manieren roteren, afhankelijk van de manier waarop u de rotatie-eigenschappen van een object instelt: `rotationX`, `rotationY` en `rotationZ`.

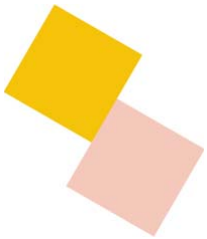
In de afbeelding hieronder ziet u twee vierkanten die niet zijn geroteerd:

In de volgende afbeelding ziet u de twee vierkanten nadat de eigenschap `rotationY` van de container van de vierkanten is verhoogd om de vierkanten te roteren langs de y-as. Door de container, of het bovenliggende weergaveobject, van de twee vierkanten te roteren, worden de beide vierkanten geroteerd:

```
container.rotationY += 10;
```

In de volgende afbeelding ziet u wat er gebeurt als u de eigenschap `rotationX` van de container voor de vierkanten instelt. Hierdoor worden de vierkanten geroteerd langs de x-as.

In de volgende afbeelding ziet u wat er gebeurt als u de eigenschap `rotationZ` van de container van de vierkanten verhoogt. Hierdoor roteren de vierkanten langs de z-as.



Een weergaveobject kan tegelijkertijd worden verplaatst en geroteerd in de 3D-ruimte.

3D-objecten projecteren op een 2D-weergave

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De klasse PerspectiveProjection in het pakket `flash.geom` biedt een eenvoudige manier om rudimentair perspectief toe te passen bij het verplaatsen van weergaveobjecten in en 3D-ruimte.

Als u niet expliciet een perspectiefprojectie maakt voor de 3D-ruimte, gebruikt de 3D-engine een standaard PerspectiveProjection-object dat zich op het hoofdniveau bevindt en aan alle onderliggende items wordt doorgegeven.

De volgende drie eigenschappen leggen vast hoe een PerspectiveProjection-object de 3D-ruimte weergeeft:

- `fieldOfView`
- `projectionCenter`
- `focalLength`

Als u de waarde van `fieldOfView` wijzigt, wordt automatisch de waarde van `focalLength` gewijzigd en andersom, omdat de eigenschappen onderling afhankelijk zijn.

De waarde van `focalLength` wordt bij een bepaalde waarde van `fieldOfView` berekend met de volgende formule:

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

Over het algemeen wordt de eigenschap `fieldOfView` alleen expliciet gewijzigd.

Gezichtsveld

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Door de eigenschap `fieldOfView` van de klasse PerspectiveProjection te bewerken, kunt u een 3D-weergaveobject maken dat groter lijkt te worden als het dichterbij de kijker komt, en kleiner lijkt te worden als het verder weg van de kijker gaat.

Met de eigenschap `fieldOfView` legt u een hoek tussen 0 en 180 graden vast die de kracht van de perspectiefprojectie bepaalt. Hoe groter de waarde, des te sterker de vervorming die wordt toegepast op een weergaveobject dat wordt verplaatst langs de z-as. Een lage waarde van `fieldOfView` leidt tot zeer weinig schaling, waardoor objecten slechts een klein stukje terug lijken te bewegen in de ruimte. Een hoge waarde van `fieldOfView` leidt tot meer vervorming, waardoor de beweging groter lijkt. De maximumwaarde van 179,9999... graden resulteert in een extreem bolvormig lenseffect. De maximumwaarde van `fieldOfView` is 179,9999... en de minimumwaarde is 0,00001... Precies 0 en 180 zijn ongeldige waarden.

Projectiemiddelpunt

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De eigenschap `projectionCenter` representeert het verdwijnpunt in de perspectiefprojectie. De eigenschap wordt toegepast als een verschuiving ten opzichte van het standaardregistratiepunt (0,0) in de linkerbovenhoek van het werkgebied.

Een object dat zich van de kijker af lijkt te verplaatsen, wordt scheefgetrokken in de richting van het verdwijnpunt en verdwijnt uiteindelijk. Stelt u zich een oneindig lange gang voor. Als u naar het einde van de gang kijkt, lijken de randen van de wanden samen te komen in een verdwijnpunt ver achter in de gang.

Als het verdwijnpunt zich in het middelpunt van het werkgebied bevindt, lijkt de gang te verdwijnen in een punt in het middelpunt. De standaardwaarde voor de eigenschap `projectionCenter` is het midden van het werkgebied. Als u bijvoorbeeld wilt dat elementen verschijnen aan de linkerzijde van het werkgebied en dat een 3D-gebied verschijnt aan de rechterzijde, stelt u `projectionCenter` in op een punt rechts van het werkgebied om dat punt het verdwijnpunt van het 3D-weergavegebied te maken.

Brandpuntsafstand

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De eigenschap `focalLength` representeert de afstand tussen de oorsprong van het gezichtspunt (0,0,0) en de locatie van het weergaveobject op de z-as.

Een lange brandpuntsafstand is net als een telelens met een smalle weergave en gecomprimeerde afstanden tussen objecten. Een korte brandpuntsafstand is vergelijkbaar met een groothoeklens waarmee u een breed beeld en veel vervorming krijgt. Een gemiddelde brandpuntsafstand benadert wat het menselijk oog ziet.

De `focalLength` wordt meestal dynamisch opnieuw berekend tijdens perspectieftransformatie terwijl het weergaveobject beweegt, maar u kunt de brandpuntsafstand ook expliciet instellen.

Standaardwaarden voor perspectiefprojectie

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het standaard PerspectiveProjection-object op het hoofdniveau heeft de volgende waarden:

- `fieldOfView: 55`
- `perspectiveCenter: stagewidth/2, stageHeight/2`
- `focalLength: stageWidth/ 2 * ( cos(fieldOfView/2) / sin(fieldOfView/2) )`

Deze waarden wordt gebruikt als u niet een eigen PerspectiveProjection-object maakt.

U kunt een eigen PerspectiveProjection-object instantiëren om `projectionCenter` en `fieldOfView` zelf te wijzigen. In dit geval heeft het gemaakte nieuwe object de volgende standaardwaarden, gebaseerd op een werkgebied met een standaardgrootte van 500 x 500:

- `fieldOfView: 55`
- `perspectiveCenter: 250,250`
- `focalLength: 480.24554443359375`

Voorbeeld: Perspectiefprojectie

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In het volgende voorbeeld ziet u hoe u perspectiefprojectie gebruikt om 3D-ruimte te maken. U ziet hoe u het verdwijnpunt kunt aanpassen en de perspectiefprojectie van de ruimte kunt wijzigen met de eigenschap `projectionCenter`. Door deze wijziging wordt afgedwongen dat `focalLength` en `fieldOfView` opnieuw worden berekend met de vervorming van de 3D-ruimte die hiermee gepaard gaat.

In dit voorbeeld gebeurt het volgende:

1. Er wordt een sprite gemaakt met de naam `center`, als een cirkel met kruisdraden.
2. De coördinaten van de sprite `center` worden toegewezen aan de eigenschap `projectionCenter` van de eigenschap `perspectiveProjection` van de eigenschap `transform` van het hoofdniveau.
3. Er worden gebeurtenislisteners voor muisgebeurtenissen toegevoegd die `projectionCenter` wijzigen zodat deze de locatie van het `center`-object volgt.
4. Er worden vier vakken in accordeonstijl gemaakt die de wanden vormen van de respectieve ruimte.

Sleep de cirkel naar verschillende locaties wanneer u dit voorbeeld, ProjectionDragger.swf, test. Het verdwijnpunt volgt de cirkel en landt waar u het neerzet. Kijk hoe de vakken die de ruimte omsluiten, worden uitgerekt en vervormen als u het projectiemiddelpunt ver van het midden van het werkgebied verplaatst.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de toepassingsbestanden van ProjectionDragger in de map 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;
        }
    }
}
```

Gebruik de klasse Matrix3D voor complexere perspectiefprojecties.

Complexe 3D-transformaties uitvoeren

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de klasse Matrix3D kunt u 3D-punten in een coördinaatruimte transformeren of 3D-punten van een coördinaatruimte toewijzen aan een andere coördinaatruimte.

U hoeft de wiskunde achter de matrix niet te begrijpen om de klasse Matrix3D te gebruiken. De meeste algemene transformatiebewerkingen kunnen worden afgehandeld met de methoden van de klasse. U hoeft zich geen zorgen te maken over het expliciet instellen of berekenen van de waarden van elk element in de matrix.

Nadat u de eigenschap `z` van een weergaveobject hebt ingesteld op een numerieke waarde, kunt u de transformatiematrix van het object ophalen met de eigenschap Matrix3D van het Transform-object van het weergaveobject:

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

U kunt de methoden van het Matrix3D-object gebruiken om translatie, rotatie, schaling en perspectiefprojectie toe te passen op het weergaveobject.

Gebruik de klasse Vector3D met de eigenschappen `x`, `y` en `z` voor het beheer van 3D-punten. De klasse kan ook een ruimtelijke vector in fysica representeren met een richting en een magnitude. Met de methoden van de klasse Vector3D kunt u algemene berekeningen met ruimtelijke vectoren uitvoeren, zoals optellen, scalair product en kruisproduct.

Opmerking: *De klasse Vector3D is niet gerelateerd aan de ActionScript-klasse Vector. De klasse Vector3D bevat eigenschappen en methoden voor het vastleggen en bewerken van 3D-punten, terwijl de klasse Vector arrays van objecten met een bepaald type ondersteunt.*

Matrix3D-objecten maken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Er zijn drie manieren om `Matrix3D`-objecten te maken of op te halen:

- Gebruik de constructormethode `Matrix3D()` om een nieuwe matrix te instantiëren. De constructor `Matrix3D()` neemt een `Vector`-object dat 16 numerieke waarden bevat, en plaatst elke waarde in een cel van de matrix. Bijvoorbeeld:

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- Stel de eigenschap `z` van een weergaveobject in. Haal vervolgens de transformatiematrix van de eigenschap `transform.matrix3D` van dat object op.
- Haal het Matrix3D-object op dat de weergave van 3D-objecten op het werkgebied regelt door de `perspectiveProjection.toMatrix3D()`-methode van het hoofdweergaveobject aan te roepen.

Meerdere 3D-transformaties toepassen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een groot aantal 3D-transformaties tegelijk toepassen met behulp van een Matrix3D-object. U wilt bijvoorbeeld een kubus roteren, schalen en vervolgens verplaatsen. U kunt dan drie afzonderlijke transformaties toepassen op elk punt van de kubus. Het is echter efficiënter om meerdere transformaties vooraf te berekenen in één Matrix3D-object en vervolgens één matrixtransformatie uit te voeren op elk punt.

Opmerking: *De volgorde waarin matrixtransformaties worden toegepast, is van belang. Matrixberekeningen zijn niet verwisselbaar. Een rotatie gevolgd door een translatie, heeft bijvoorbeeld een ander resultaat dan dezelfde translatie gevolgd door dezelfde rotatie.*

In het volgende voorbeeld worden twee manieren getoond om meerdere 3D-transformaties uit te voeren.

```
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;
        }
    }
}
```

In de methode `doTransforms()` gebruikt het eerste codeblok de DisplayObject-eigenschappen om de rotatie, schaling en positie van een rechthoekige vorm te wijzigen. Het tweede codeblok gebruikt de methoden van de klasse Matrix3D om dezelfde transformaties uit te voeren.

Het grootste voordeel van de methoden van `Matrix3D` is dat alle berekeningen eerst in de matrix worden uitgevoerd. Vervolgens worden ze maar één keer op het weergaveobject toegepast wanneer de eigenschap `transform.matrix3D` wordt ingesteld. Door het instellen van de DisplayObject-eigenschappen wordt de broncode iets eenvoudiger om te lezen. Maar elke keer dat een rotatie- of schalingseigenschap wordt ingesteld, moeten er meerdere berekeningen worden uitgevoerd en weergaveobjecteigenschappen worden gewijzigd.

Als uw code dezelfde complexe transformaties meerdere keren toepast op weergaveobjecten, kunt u het Matrix3D-object als variabele opslaan en vervolgens zo vaak als nodig opnieuw toepassen.

Matrix3D-objecten gebruiken om de weergavevolgorde te wijzigen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Zoals eerder vermeld, bepaalt de laagvolgorde van weergaveobjecten in de weergavelijst de volgorde waarin lagen worden weergegeven, ongeacht hun relatieve z-as. Als uw animatie de eigenschappen van weergaveobjecten verandert in een volgorde die afwijkt van de volgorde in de weergavelijst, ziet de kijker mogelijk een laagwerking van weergaveobjecten die niet overeenkomt met de laagwerking van de z-as. Een object dat ver weg van de kijker moet lijken, wordt dan mogelijk weergegeven vóór een object dat dichterbij de kijker is.

Als u zeker wilt weten dat de laagvolgorde van 3D-weergaveobjecten overeenkomt met de relatieve diepte van de objecten, gebruikt u een werkwijze als de volgende:

1. Haal met de methode `getRelativeMatrix3D()` van het Transform-object de relatieve `z-assen` van de onderliggende 3D-weergaveobjecten op.
2. Verwijder met de methode `removeChild()` de objecten uit de weergavelijst.
3. Sorteer de weergaveobjecten op basis van de relatieve waarde van de z-as.
4. Voeg met de methode `addChild()` de onderliggende objecten in omgekeerde volgorde weer toe aan de weergavelijst.

Deze nieuwe volgorde zorgt dat de objecten worden weergegeven in overeenstemming met de relatieve z-as.

De volgende code dwingt de correcte weergave van de zes vlakken van een 3D-vak af. De vlakken van het vak worden opnieuw geordend nadat er rotaties op zijn toegepast:

```
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);
    }
}
```

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de toepassingsbestanden in de map Samples/ReorderByZ.

Driehoeken gebruiken voor 3D-effecten

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In ActionScript kunt u bitmaptransformaties uitvoeren met de methode `Graphics.drawTriangles()`, omdat 3D-modellen worden gerepresenteerd door een verzameling driehoeken in de ruimte. Flash Player en AIR ondersteunen echter geen dieptebuffer zodat weergaveobjecten zelf vlak, of 2D, zijn. Dit wordt beschreven in "Inzicht in 3D-weergaveobjecten in Flash Player en de AIR-runtime" op pagina 371.) De methode `Graphics.drawTriangles()` is vergelijkbaar met de methode `Graphics.drawPath()` omdat de methode een set coördinaten neemt om een driehoekpad te tekenen.

Zie voor meer informatie over het gebruik van `Graphics.drawPath()`, "Tekenpaden" op pagina 249.

De methode `Graphics.drawTriangles()` gebruikt een Vector.<Number> om de puntlocaties op te geven voor het driehoekpad:

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number>
= null, culling:String = "none"):void
```

De eerste parameter van `drawTriangles()` is de enige vereiste parameter: de parameter `vertices`. Deze parameter is een vector van getallen die de coördinaten definiëren waarmee de driehoeken worden getekend. Elke drie sets coördinaten (zes getallen) representeren een driehoekpad. Zonder de parameter `indices` zou de lengte van de vector altijd een factor van zes zijn omdat er voor elke driehoek drie coördinatenparen nodig zijn (drie sets van twee x/y-waarden). Bijvoorbeeld:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

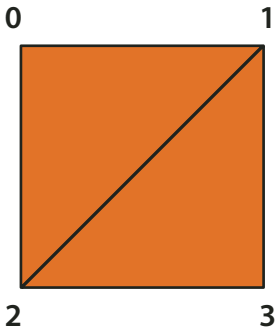
Geen van deze driehoeken heeft een gemeenschappelijk punt. Maar als dit wel het geval was, zou de tweede parameter van `drawTriangles()` - `indices` - kunnen worden gebruikt om waarden in de vector `vertices` opnieuw te gebruiken voor meer dan één driehoek.

Houd er bij het gebruik van de parameter `indices` rekening mee dat de waarden van `indices` puntindices zijn en geen indices die direct zijn gerelateerd aan de elementen van de array `vertices`. Met andere woorden: een index in de vector `vertices` zoals gedefinieerd door `indices`, is in feite de werkelijke index gedeeld door 2. Voor het derde punt van een vector `vertices` gebruikt u bijvoorbeeld de waarde 2 voor `indices`, ook al begint de eerste numerieke waarde van dat punt bij vectorindex 4.

In dit voorbeeld worden twee driehoeken samengevoegd met de parameter `indices` zodat ze dezelfde diagonale rand hebben:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

Zoals u ziet, wordt er een vierkant getekend met behulp van twee driehoeken, maar zijn er slechts vier punten opgegeven in de vector `vertices`. Met behulp van `indices` worden de twee punten die worden gedeeld door de twee driehoeken, opnieuw gebruikt voor elke driehoek. Hierdoor wordt het aantal vertices teruggebracht van 6 (12 getallen) naar 4 (8 getallen):



Een vierkant getekend met twee driehoeken met behulp van de parameter vertices

Deze techniek is nuttig bij grotere driehoeknetten waarbij de meeste punten worden gedeeld door meerdere driehoeken.

Alle vullingen kunnen worden toegepast op driehoeken. De vullingen worden op dezelfde manier toegepast op het resulterende driehoeknet als op een andere vorm.

Bitmaps transformeren

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Bitmaptransformaties wekken de indruk van perspectief of "structuur" op een driedimensionaal object. Hiermee kunt u een bitmap vervormen in de richting van het verdwijnpunt waardoor de afbeelding steeds kleiner lijkt te worden als deze dichterbij het verdwijnpunt komt. Of u kunt een tweedimensionale bitmap gebruiken om een oppervlak te maken voor een driedimensionaal object, waarbij u de illusie wekt van structuur of een “wikkel” op dat driedimensionale object.

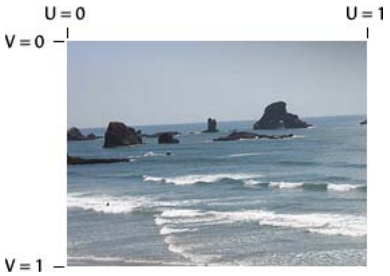
Een tweedimensionaal oppervlak met een verdwijnpunt en een driedimensionaal object waar overheen een bitmap is gewikkeld.

UV-toewijzing

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

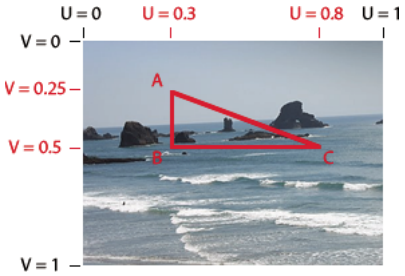
Als u begint te werken met structuren, wilt u waarschijnlijk gebruikmaken van de parameter uvtData van `drawTriangles()`. Met deze parameter kunt u UV-toewijzing instellen voor bitmapvullingen.

UV-toewijzing is een methode voor het geven van structuur aan objecten. De methode is gebaseerd op twee waarden: een horizontale (x) U-waarde en een verticale (y) V-waarde. De waarden zijn niet gebaseerd op pixelwaarden, maar op percentages. 0 U en 0 V is helemaal linksboven in een afbeelding en 1 U en 1 V is helemaal rechtsonder:



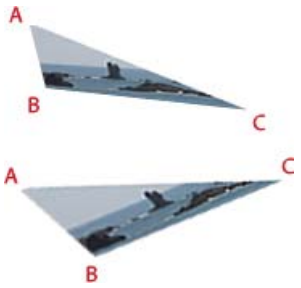
De locaties UV 0 en 1 op een bitmapafbeelding

U kunt vectoren van een driehoek UV-coördinaten geven om deze zelf te koppelen aan de respectieve locaties op een afbeelding:



De UV-coördinaten van een driehoekig gebied van een bitmapafbeelding

De UV-waarden blijven consistent met de punten van de driehoek:



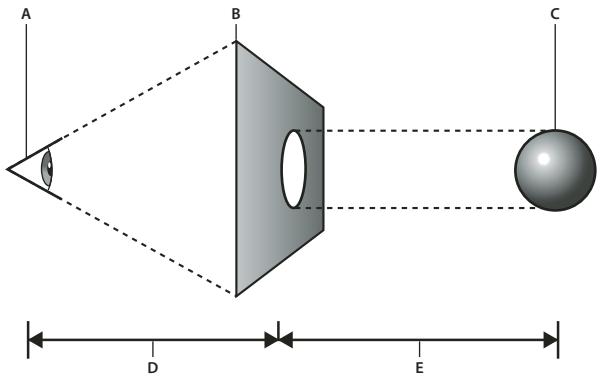
De hoekpunten van de driehoek worden verplaatst en de bitmap wordt vervormd om te zorgen dat de UV-waarden voor een individueel punt gelijk blijven

Aangezien 3D-transformaties in ActionScript worden toegepast op de driehoek die is gekoppeld aan de bitmap, wordt de bitmapafbeelding op basis van de UV-waarden toegepast op de driehoek. Dus in plaats van matrixberekeningen te gebruiken, kunt u de UV-waarden instellen of aanpassen om een driedimensionaal effect te creëren.

De methode `Graphics.drawTriangles()` accepteert ook optionele informatie voor driedimensionale transformaties: de T-waarde. De T-waarde in uvtData vertegenwoordigt het 3D-perspectief, of specifieker gezegd, de schaalfactor van het toegewezen hoekpunt (vertex). UVT-toewijzing voegt perspectiefcorrectie toe aan UV-toewijzing. Als een object bijvoorbeeld in de 3D-ruimte op afstand van het gezichtspunt is geplaatst zodat het 50% van de "oorspronkelijke" grootte lijkt te zijn, is de T-waarde van dat object 0.5. Aangezien driehoeken worden getekend om objecten in de 3D-ruimte te vertegenwoordigen, bepalen de locaties langs de z-as hun T-waarden. De T-waarde wordt bepaald met de volgende vergelijking:

```
T = focalLength/(focalLength + z);
```

In de vergelijking staat focalLength voor een brandpuntsafstand of een berekende locatie op het scherm die bepaalt hoeveel perspectief wordt toegevoegd aan de weergave.



De focuslengte en z-waarde
***A.** gezichtspunt **B.** scherm **C.** 3D-object **D.** waarde van focalLength **E.** z-waarde*

Met behulp van de waarde van T worden basisvormen geschaald zodat ze verder weg lijken. Het is in het algemeen de waarde die wordt gebruikt om 3D-punten om te zetten in 2D-punten. Bij UVT-gegevens wordt de T-waarde ook gebruikt om een bitmap te schalen tussen de punten in een driehoek met perspectief.

Wanneer u UVT-waarden definieert, volgt de T-waarde direct de UV-waarden die zijn gedefinieerd voor een hoekpunt. Door de toevoeging van T komen alle drie waarden in de parameter `uvtData` (U, V en T) overeen met alle twee waarden in de parameter `vertices` (x en y). Met alleen UV-waarden geldt uvtData.length == vertices.length. Met de opname van een T-waarde geldt uvtData.length = 1,5*vertices.length.

In het volgende voorbeeld wordt een vlak getoond dat in de 3D-ruimte wordt geroteerd met behulp van UVT-gegevens. Het voorbeeld gebruikt de afbeelding ocean.jpg en de "helperklasse" ImageLoader, die de afbeelding ocean.jpg laadt zodat deze kan worden toegewezen aan het BitmapData-object.

Hier is de bron van de klasse ImageLoader. Sla deze code op in een bestand met de naam ImageLoader.as:

```
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);
    }
    }
}
```

En hier is het ActionScript waarin driehoeken, UV-toewijzing en T-waarden worden gebruikt om de afbeelding kleiner te laten worden in de richting van een verdwijnpunt en te laten roteren. Sla deze code op in een bestand met de naam 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
```

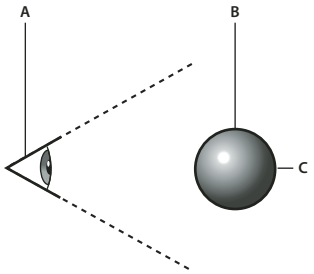
```
            t1 = focalLength/(focalLength + z1);
            t2 = focalLength/(focalLength + z2);
            t3 = focalLength/(focalLength + z3);
            t4 = focalLength/(focalLength + z4);

            // 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);
        }
    }
}
```

U kunt dit voorbeeld testen door deze twee klassenbestanden op te slaan in dezelfde map als de afbeelding "ocean.jpg". U kunt zien hoe de oorspronkelijke bitmap wordt getransformeerd zodat het lijkt of de bitmap verdwijnt in de verte en roteert in de 3D-ruimte.

Schifting

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Schifting is het proces waarin wordt bepaald welke vlakken van een driedimensionaal object de renderer niet hoeft te renderen omdat ze vanuit het huidige gezichtspunt niet zichtbaar zijn. In de 3D-ruimte is het oppervlak aan de "achterzijde" van een driedimensionaal object verborgen vanuit het gezichtspunt.



De achterzijde van een 3D-object is niet zichtbaar vanuit het gezichtspunt.
***A.** gezichtspunt **B.** 3D-object **C.** de achterzijde van een driedimensionaal object*

Driehoeken worden altijd gerenderd, ongeacht de grootte, vorm of positie. Schifting zorgt dat Flash Player of AIR het 3D-object correct rendert. Daarnaast wilt u soms dat de renderer sommige driehoeken overslaat zodat er minder rendercycli nodig zijn. Denk aan een kubus die roteert in de ruimte. U ziet nooit meer dan drie zijden van de kubus. Er zijn altijd drie zijden uit zicht die de andere kant van de kubus op wijzen. Aangezien deze zijden niet zichtbaar zijn, hoeft de renderer ze niet te tekenen. Zonder schifting rendert Flash Player of AIR zowel de zijden aan de voor- als aan de achterkant.

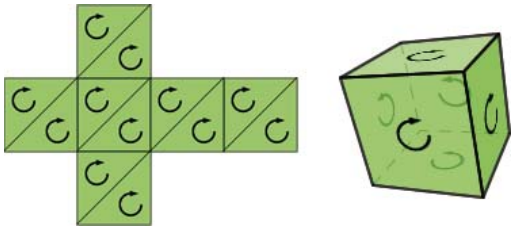
Een kubus heeft zijden die niet zichtbaar zijn vanuit het huidige gezichtspunt.

Daarom heeft de methode `Graphics.drawTriangles()` een vierde parameter om een schiftingswaarde in te stellen:

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

De parameter culling is een waarde van de opsommingsklasse `TriangleCulling`: `TriangleCulling.NONE`, `TriangleCulling.POSITIVE` en `TriangleCulling.NEGATIVE`. Deze waarden zijn afhankelijk van de richting van het driehoekpad dat het oppervlak van het object definieert. De ActionScript-API voor het bepalen van de schifting veronderstelt dat alle naar buiten wijzende driehoeken van een 3D-vorm worden getekend met dezelfde padrichting. Nadat een driehoek rond heeft gedraaid, verandert ook de padrichting. Op dat moment kan de driehoek worden geschift, wat betekent dat de driehoek niet meer wordt gerenderd.

Dus met voor `TriangleCulling` de waarde `POSITIVE` verwijdert u driehoeken met een positieve padrichting (rechtsom) uit het renderproces. Met voor `TriangleCulling` de waarde `NEGATIVE` verwijdert u driehoeken met een negatieve padrichting (linksom) uit het renderproces. In het geval van een kubus hebben de oppervlakken aan de voorzijde een positieve padrichting en hebben de oppervlakken aan de achterzijde een negatieve padrichting:



De padrichting is zichtbaar op een uitgeklapte kubus. Ingeklapt is de padrichting aan de achterzijde omgekeerd.

Als u wilt zien hoe schifting werkt, begint u met het eerdere voorbeeld voor "UV-toewijzing" op pagina 385 en stelt u de parameter culling van de methode `drawTriangles()` in op `TriangleCulling.NEGATIVE`:

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

Zoals u ziet, wordt de achterzijde van de afbeelding niet gerenderd terwijl het object roteert.

Hoofdstuk 20: Basisbeginselen van het werken met tekst

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u tekst op het scherm wilt weergeven in Adobe® Flash® Player of Adobe® AIR™, gebruikt u een instantie van de klasse TextField of gebruikt u de Flash Text Engine-klassen. Met deze klassen kunt u tekst maken, weergeven en opmaken. U kunt desgewenst ook het Text Layout Framework (TLF) gebruiken, een componentenbibliotheek die is gebaseerd op de Flash Text Engine-klassen, maar die speciaal is ontworpen voor gebruiksgemak. Bij mobiele apparaten kunt u de klasse StageText gebruiken voor tekstinvoer.

U kunt specifieke inhoud opstellen voor tekstvelden of de bron van de tekst aangeven en vervolgens instellen hoe tekst eruit moet zien. U kunt ook reageren op gebruikersgebeurtenissen wanneer de gebruiker tekst invoert of op een hypertextkoppeling klikt.

Met zowel de TextField-klasse als de Flash Text Engine-klassen kunt tekst in Flash Player en AIR weergeven en beheren. U kunt de TextField-klasse gebruiken voor het maken van tekstobjecten voor weergave en invoer. De klasse TextField biedt de basis voor andere op tekst gebaseerde componenten, zoals TextArea en TextInput. Met de klasse TextFormat kunt u teken- en alineaopmaak instellen voor TextField-objecten en kunt u CSS (Cascading Style Sheets) toepassen met behulp van de eigenschap Textfield.styleSheet en de klasse StyleSheet. U kunt tekst met HTML-opmaak, die ingesloten media (filmclips, SWF-bestanden, GIF-bestanden, PNG-bestanden en JPEG-bestanden), rechtstreeks aan een tekstveld toewijzen.

De Flash Text Engine, die beschikbaar is vanaf Flash Player 10 en Adobe AIR 1.5, biedt ondersteuning op een laag niveau voor geavanceerde besturing van tekstmeeteenheden, opmaak en bidirectionele tekst. De Flash Text Engine biedt ook verbeterde tekstdoorloop en uitgebreide taalondersteuning. Hoewel u de Flash Text Engine kunt gebruiken voor het maken en beheren van tekstelementen, is deze primair bedoeld voor het maken van tekstafhandelingscomponenten en is hiervoor een grotere programmeerkennis voor nodig. Het Text Layout Framework, dat beschikt over een tekstafhandelingscomponent op basis van de Flash Text Engine, biedt een eenvoudigere mogelijkheid om de uitgebreide functies van de nieuwe tekstengine te gebruiken. Het Text Layout Framework is een uitbreidbare bibliotheek die geheel in ActionScript 3.0 is gemaakt. U kunt gebruik maken van de bestaande TLF-component of van het framework om uw eigen tekstcomponent te maken.

De klasse StageText is beschikbaar vanaf AIR 3 en biedt een native tekstinvoerveld. Aangezien dit veld wordt aangeleverd door het besturingssysteem van het apparaat, zijn de gebruikers van het apparaat bekend met de functionaliteit en de vormgeving ervan. Een StageText-instantie is geen weergaveobject. In plaats van de instantie toe te voegen aan de weergavelijst, kunt u een werkgebied en een weergavegebied aan een instantie toewijzen. Dit weergavegebied bevindt zich op het werkgebied en wordt ook wel een 'viewport' genoemd. De StageText-instantie wordt vóór de weergaveobjecten weergegeven.

Hier vindt u meer informatie over deze onderwerpen:

- “De klasse TextField gebruiken” op pagina 393
- “De Flash Text Engine gebruiken” op pagina 418
- “Text Layout Framework gebruiken” op pagina 448
- Native tekstinvoer met StageText

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot tekstafhandeling:

Trapsgewijze opmaakmodellen Een standaardsyntaxis voor het opgeven van stijlen en opmaak voor inhoud die is gestructureerd in de indeling XML (of HTML).

Apparaatlettertype Een lettertype dat is geïnstalleerd op het apparaat van de gebruiker.

Dynamisch tekstveld Een tekstveld waarvan de inhoud kan worden gewijzigd door ActionScript, maar niet door invoer van de gebruiker.

Ingesloten lettertype Een lettertype waarvan de tekenomtrekgegevens zijn opgeslagen in het SWF-bestand van de toepassing.

HTML-tekst Tekstinhoud die wordt ingevoerd in een tekstveld met behulp van ActionScript en die naast de eigenlijke tekstinhoud HTML-opmaaktags bevat.

Invoertekstveld Een tekstveld waarvan de inhoud kan worden gewijzigd door de gebruiker of door ActionScript.

Tekenspatiëring Een aanpassing van de tussenruimte tussen tekenparen om de tussenruimte in woorden meer proportioneel te maken en de tekst leesbaarder te maken.

Statisch tekstveld Een tekstveld dat is gemaakt met het ontwerpgereedschap, waarvan de inhoud niet kan worden gewijzigd wanneer het SWF-bestand wordt uitgevoerd.

Tekstregelmeeteenheden Meeteenheden van de grootte van verschillende delen van de tekstinhoud in een tekstveld, zoals de basislijn van de tekst, de hoogte van de bovenzijde van de tekens, de grootte van de staarten (het deel van sommige kleine letters dat onder de basislijn uitsteekt), enzovoort.

Tekstspatiëring Aanpassing van de ruimte tussen groepen letters of blokken tekst om de dichtheid te verhogen of verlagen en de tekst leesbaarder te maken.

Hoofdstuk 21: De klasse TextField gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een instantie van de klasse TextField gebruiken om tekst weer te geven of een tekstinvoerveld op het scherm te maken in Adobe® Flash® Player of Adobe® AIR™. De klasse TextField is de basis voor andere op tekst gebaseerde componenten, zoals TextArea of TextInput.

De inhoud van een tekstveld kan vooraf worden opgegeven in het SWF-bestand, worden geladen uit een tekstbestand of database, of worden ingevoerd door een gebruiker die met uw toepassing werkt. In een tekstveld kan de tekst verschijnen als gerenderde HTML-inhoud, met afbeeldingen die zijn ingesloten in de gerenderde HTML. Nadat u een instantie van een tekstveld hebt gemaakt, kunt u flash.text-klassen gebruiken, zoals TextFormat en StyleSheet, om het uiterlijk van de tekst te bepalen. Het flash.text-pakket bevat vrijwel alle klassen die te maken hebben met het maken, beheren en opmaken van tekst in ActionScript.

U kunt tekst opmaken door de opmaak te definiëren met een TextFormat-object en dat object aan het tekstveld toe te wijzen. Als uw tekstveld HTML-tekst bevat, kunt u een StyleSheet-object toepassen op het tekstveld om stijlen toe te wijzen aan specifieke stukken tekstveldinhoud. Het TextFormat-object of StyleSheet-object bevat eigenschappen die de weergave van de tekst definiëren, zoals kleur, grootte en dikte. Het TextFormat-object wijst de eigenschappen toe aan alle inhoud binnen een tekstveld of een tekstbereik. Binnen hetzelfde tekstveld kan bijvoorbeeld één zin vette rode tekst bevatten en de volgende zin blauwe cursieve tekst.

Naast de klassen in het flash.text-pakket kunt u met de klasse flash.events.TextEvent reageren op acties van de gebruiker met betrekking tot tekst.

Meer Help-onderwerpen

"Tekstopmaak toewijzen" op pagina 401

"HTML-tekst weergeven" op pagina 395

"Trapsgewijze opmaakmodellen toepassen" op pagina 402

Tekst weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Hoewel ontwerpgereedschappen zoals Adobe Flex Builder en Flash Professional verschillende opties bieden voor tekstweergave, inclusief tekstgerelateerde componenten of tekstgereedschappen, is weergave via een tekstveld de eenvoudigste wijze om tekst via programmacode weer te geven.

Typen tekst

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het type tekst in een tekstveld wordt bepaald door de bron ervan:

- Dynamische tekst

  Dynamische tekst is inhoud die wordt geladen van een externe bron, zoals een tekstbestand, een XML-bestand of zelfs een externe webservice.

- Invoertekst

  Invoertekst is alle tekst die wordt ingevoerd door een gebruiker of dynamische tekst die een gebruiker kan bewerken. U kunt een opmaakmodel opzetten om invoertekst op te maken of u kunt de klasse flash.text.TextFormat gebruiken om eigenschappen toe te wijzen aan het tekstveld voor de invoerinhoud. Zie "Tekstinvoer vastleggen" op pagina 399 voor meer informatie.

- Statische tekst

  Statische tekst wordt alleen via Flash Professional gemaakt. U kunt geen instantie van statische tekst maken met behulp van ActionScript 3.0. U kunt echter wel ActionScript-klassen als StaticText en TextSnapshot gebruiken om een bestaande instantie van statische tekst te manipuleren. Zie "Werken met statische tekst" op pagina 407 voor meer informatie.

Inhoud van tekstvelden wijzigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt dynamische tekst definiëren door een string toe te wijzen aan de eigenschap `flash.text.TextField.text`. U wijst als volgt een tekenreeks direct toe aan de eigenschap:

```
myTextField.text = "Hello World";
```

U kunt ook aan de eigenschap `text` een waarde toewijzen uit een variabele die in uw script is gedefinieerd, zoals in het volgende voorbeeld:

```
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;
        }
    }
}
```

Als alternatief kunt u aan de eigenschap `text` een waarde toewijzen uit een externe variabele. U hebt drie opties voor het laden van tekstwaarden uit externe bronnen:

- Met de klassen flash.net.URLLoader en flash.net.URLRequest laadt u variabelen voor de tekst vanuit een lokale of externe locatie.
- Het kenmerk `FlashVars` is ingesloten in de HTML-pagina die als host dient voor het SWF-bestand en kan waarden bevatten voor tekstvariabelen.
- Met de klasse flash.net.SharedObject beheert u de permanente opslag van waarden. Zie "Lokale gegevens opslaan" op pagina 741 voor meer informatie.

HTML-tekst weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse flash.text.TextField heeft een eigenschap `htmlText` waarmee u uw tekstreeks kunt identificeren als een reeks die HTML-tags bevat voor het opmaken van de inhoud. Zoals te zien is in het volgende voorbeeld, moet u de tekenreekswaarde toewijzen aan de eigenschap `htmlText` (niet de eigenschap `text`) om Flash Player of AIR de tekst te laten weergeven als HTML:

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as <u>HTML</u> text.</p>";
myTextBox.htmlText = myText;
```

Flash Player en AIR ondersteunen een subset van de HTML-tags en -entiteiten voor de eigenschap `htmlText`. De eigenschapsbeschrijving `flash.text.TextField.htmlText` in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform biedt gedetailleerde informatie over de ondersteunde HTML-tags en -entiteiten.

Wanneer u de inhoud hebt aangegeven met de eigenschap `htmlText`, kunt u met stijlpagina's of de tag `textformat` de opmaak van de inhoud beheren. Zie "Tekst opmaken" op pagina 401 voor meer informatie.

Afbeeldingen gebruiken in tekstvelden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een ander voordeel van het weergeven van inhoud als HTML-tekst is dat u afbeeldingen in het tekstveld kunt opnemen. U kunt naar een lokale of externe afbeelding verwijzen met de tag `img` en de afbeelding laten verschijnen in het betreffende tekstveld.

In het volgende voorbeeld wordt een tekstveld met de naam `myTextBox` gemaakt en wordt in de weergegeven tekst een JPG-afbeelding van een oog opgenomen die in dezelfde map is opgeslagen als het SWF-bestand:

```
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;
        }
    }
}
```

De tag `img` ondersteunt JPEG-, GIF-, PNG- en SWF-bestanden.

Schuivende tekst in een tekstveld

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In veel gevallen kan de tekst langer zijn dan het tekstveld waarin de tekst wordt weergegeven. Of u hebt misschien een invoerveld waarin gebruikers tekst kunnen invoeren die langer is dan in één keer kan worden weergegeven. U kunt met behulp van de aan schuiven gerelateerde eigenschappen van de klasse flash.text.TextField lange inhoud beheren, zowel verticaal als horizontaal.

De aan schuiven gerelateerde eigenschappen omvatten `TextField.scrollV`, `TextField.scrollH` en `maxScrollV` en `maxScrollH`. Gebruik deze eigenschappen om te reageren op gebeurtenissen, zoals een muisklik of een druk op een toets.

In het volgende voorbeeld wordt een tekstveld gemaakt dat een setgrootte is en meer tekst bevat dan het veld in één keer kan weergeven. Als de gebruiker op het tekstveld klikt, wordt de tekst verticaal geschoven.

```
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++;
        }
    }
}
```

Tekst selecteren en manipuleren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt dynamische of invoertekst selecteren. Omdat de tekstselectie-eigenschappen en -methoden van de klasse TextField gebruik maken van de indexposities om het tekstbereik in te stellen dat moet worden gemanipuleerd, kunt u programmatisch dynamische of invoertekst selecteren, ook al kent u de inhoud niet.

Opmerking: *Wanneer u in Flash Professional de selecteerbare optie voor een statisch tekstveld kiest, is het geëxporteerde en in het weergaveoverzicht geplaatste tekstveld een normaal, dynamische tekstveld.*

Tekst selecteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschap `flash.text.TextField.selectable` is standaard `true` en u kunt programmatisch tekst selecteren met de methode `setSelection()`.

U kunt bijvoorbeeld binnen een tekstveld een specifieke tekst instellen die, als de gebruiker op het tekstveld klikt, moet worden geselecteerd:

```
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);
}
```

Op dezelfde wijze maakt u, als u wilt dat tekst in een tekstveld wordt geselecteerd wanneer de tekst voor het eerst wordt weergegeven, een gebeurtenishandlerfunctie die wordt aangeroepen wanneer het tekstveld wordt toegevoegd aan de weergavelijst.

Door de gebruiker selecteerde tekst vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De TextField-eigenschappen `selectionBeginIndex` en `selectionEndIndex`, die "alleen-lezen" zijn en dus niet kunnen worden ingesteld om programmatisch tekst te selecteren, kunnen worden gebruikt om vast te leggen wat de gebruiker op dat moment heeft geselecteerd. Daarnaast kunnen invoertekstvelden gebruik maken van de eigenschap `caretIndex`.

Met de volgende code worden bijvoorbeeld de indexwaarden van door de gebruiker geselecteerde tekst getraceerd:

```
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);
}
```

U kunt een collectie TextFormat-objecteigenschappen toepassen op de selectie om de tekstweergave te wijzigen. Zie "Tekstbereiken in een tekstveld opmaken" op pagina 404 voor meer informatie over het toepassen van een collectie TextFormat-eigenschappen op geselecteerde tekst.

Tekstinvoer vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschap `type` van een tekstveld is standaard ingesteld op `dynamic`. Als u de eigenschap `type` instelt op `input` met de klasse TextFieldType, kunt u invoer van de gebruiker verzamelen en de waarde opslaan om te worden gebruikt in andere delen van de toepassing. Invoertekstvelden zijn handig voor formulieren en alle toepassingen waarbij de gebruiker een tekstwaarde moet definiëren om elders in het programma te worden gebruikt.

De volgende code maakt bijvoorbeeld een invoertekstveld met de naam `myTextBox`. Wanneer de gebruiker tekst in het veld invoert, wordt de gebeurtenis `textInput` geactiveerd. Een gebeurtenishandler met de naam `textInputCapture` legt de tekstreeks die wordt ingevoerd vast en wijst daaraan een variabele toe. Flash Player of AIR geeft de nieuwe tekst in een ander tekstveld met de naam `myOutputBox` weer.

```
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;
        }

    }
}
```

Tekstinvoer beperken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat invoertekstvelden vaak worden gebruikt voor formulieren of dialoogvensters in toepassingen, wilt u wellicht de typen tekens die een gebruiker in een tekstveld kan invoeren, beperken of zelfs de tekst verborgen houden, bijvoorbeeld voor een wachtwoord. De klasse flash.text.TextField heeft een eigenschap `displayAsPassword` en een eigenschap `restrict` die u kunt instellen om de gebruikersinvoer te regelen.

De eigenschap `displayAsPassword` verbergt eenvoudigweg de tekst (geeft deze weer als een reeks asterisks) wanneer de gebruiker deze typt. Wanneer `displayAsPassword` wordt ingesteld op `true`, werken de opdrachten Knippen en Kopiëren en de bijbehorende sneltoetsen niet. Zoals het volgende voorbeeld aangeeft, kunt u de eigenschap `displayAsPassword` op dezelfde manier toewijzen als andere eigenschappen, zoals de achtergrond en de kleur:

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

De eigenschap `restrict` is iets gecompliceerder omdat u moet opgeven welke tekst de gebruiker in een invoertekstveld mag typen. U kunt bepaalde letters, cijfers of bereiken van letters, cijfers en tekens toestaan. De volgende code laat de gebruiker alleen hoofdletters invoeren (en geen cijfers of speciale tekens) in het tekstveld:

```
myTextBox.restrict = "A-Z";
```

ActionScript 3.0 gebruikt afbrekingsstreepjes om bereiken te definiëren en carets om uitgesloten tekens te definiëren. Zie de vermelding over de eigenschap `flash.text.TextField.restrict` in de Naslaggids voor ActionScript® 3.0 voor Adobe® Flash® Professional CS5 voor meer informatie over het definiëren van wat er in een invoertekstveld wordt beperkt.

Opmerking: *Als u de eigenschap* `flash.text.TextField.restrict` *gebruikt, worden letters met beperkingen voor hoofdletters/kleine letters in de runtime automatisch omgezet naar de juiste hoofdletters/kleine letters. Als u de eigenschap* `fl.text.TLFTextField.restrict` *gebruikt (dat wil zeggen: als u een TLF-tekstveld gebruikt), worden letters met beperkingen voor hoofdletters/kleine letters genegeerd.*

Tekst opmaken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn verscheidene opties om de weergave van tekst programmatisch op te maken. U kunt eigenschappen rechtstreeks in de TextField-instantie instellen, bijvoorbeeld de eigenschappen `TextFIeld.thickness`, `TextField.textColor` en `TextField.textHeight`.Of u kunt de inhoud van het tekstveld aangeven met behulp van de eigenschap `htmlText` en de ondersteunde HTML-tags gebruiken, zoals `b`, `i` en `u`. Maar u kunt ook TextFormat-objecten toepassen op tekstvelden die gewone tekst bevatten, of StyleSheet-objecten op tekstvelden die de eigenschap `htmlText` bevatten. Het gebruik van TextFormat- en StyleSheet-objecten biedt de meeste controle en consistentie over de weergave van tekst overal in de toepassing. U kunt een TextFormat- of StyleSheet-object definiëren en dit toepassen op veel of alle tekstvelden in de toepassing.

Tekstopmaak toewijzen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse TextFormat gebruiken om een aantal verschillende tekstweergave-eigenschappen in te stellen en deze toe te passen op de gehele inhoud van een TextField-object of op een tekstbereik.

In het volgende voorbeeld wordt één TextFormat-object toegepast op een heel TextField-object en wordt een tweede TextFormat-object toegepast op een tekstbereik binnen dat TextField-object:

```
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);
```

De methode `TextField.setTextFormat()` is alleen van invloed op tekst die al in het tekstveld wordt weergegeven. Als de inhoud in het tekstveld verandert, moet de toepassing mogelijk de methode `TextField.setTextFormat()` opnieuw aanroepen om de opmaak opnieuw toe te passen. U kunt ook de tekstveldeigenschap `defaultTextFormat` instellen om de opmaak op te geven die moet worden gebruikt voor tekst die de gebruiker invoert.

Trapsgewijze opmaakmodellen toepassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Tekstvelden kunnen platte tekst of tekst met HTML-opmaak bevatten. Platte tekst wordt opgeslagen in de eigenschap `text` van de instantie en HTML-tekst in de eigenschap `htmlText`.

U kunt CSS-stijldeclaraties gebruiken om tekststijlen te definiëren die u kunt toepassen op veel verschillende tekstvelden. CSS-stijldeclaraties kunnen worden gemaakt in de toepassingscode of bij het uitvoeren worden geladen vanuit een extern CSS-bestand.

De klasse flash.text.StyleSheet behandelt de CSS-stijlen. De klasse StyleSheet herkent een beperkte set CSS-eigenschappen. Zie de vermelding over flash.textStylesheet in de Naslaggids voor ActionScript® 3.0 voor Adobe® Flash® Professional CS5 voor een gedetailleerde lijst stijleigenschappen die door de klasse StyleSheet worden ondersteund.

Zoals het volgende voorbeeld aangeeft, kunt u CSS maken in uw code en die stijlen toepassen op HTML-tekst door gebruik te maken van een StyleSheet-object:

```
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);
```

Wanneer u een StyleSheet-object hebt gemaakt, maakt de voorbeeldcode een eenvoudig object dat een set stijldeclaratie-eigenschappen bevat. Vervolgens wordt de methode `StyleSheet.setStyle()` aangeroepen, waarmee de nieuw stijl wordt toegevoegd aan het opmaakmodel met de naam ".darkred". Vervolgens wordt de opmaakmodelopmaak toegepast door het StyleSheet-object toe te wijzen aan de tekstveldeigenschap `styleSheet`.

CSS-stijlen werken pas wanneer het opmaakmodel wordt toegepast op het TextField-object voordat de eigenschap `htmlText` wordt ingesteld.

Een tekstveld met een opmaakmodel kan niet worden bewerkt. Als u een invoertekstveld hebt en hieraan een opmaakmodel toewijst, vertoont het tekstveld de eigenschappen van het opmaakmodel, maar staat het tekstveld niet toe dat gebruikers hierin nieuwe tekst invoeren. U kunt ook de volgende ActionScript-API's niet gebruiken op een tekstveld met een hieraan toegewezen opmaakmodel:

- De methode `TextField.replaceText()`
- De methode `TextField.replaceSelectedText()`
- De eigenschap `TextField.defaultTextFormat`
- De methode `TextField.setTextFormat()`

Als aan een tekstveld een opmaakmodel is toegewezen, maar de eigenschap `TextField.styleSheet` later wordt ingesteld op `null`, voegt de inhoud van zowel de eigenschap `TextField.text` als `TextField.htmlText` tags en kenmerken toe aan de inhoud, om de opmaak uit het eerder toegewezen opmaakmodel op te nemen. Als u de oorspronkelijke eigenschap `htmlText` wilt behouden, slaat u deze op in een variabele, voordat u het opmaakmodel instelt op `null`.

Een extern CSS-bestand laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De CSS-benadering van opmaak is krachtiger wanneer u tijdens het uitvoeren CSS-informatie kunt laden uit een extern bestand. Wanneer CSS-gegevens zich buiten de toepassing zelf bevinden, kunt u de visuele stijl van tekst in de toepassing wijzigen zonder de ActionScript 3.0-broncode te wijzigen. Wanneer de toepassing in gebruik is genomen, kunt u een extern CSS-bestand wijzigen zodat het de weergave van de toepassing wijzigt zonder het SWF-bestand van de toepassing opnieuw in gebruik te hoeven nemen.

Met de methode `StyleSheet.parseCSS()` converteert u een tekenreeks die CSS-gegevens bevat in stijldeclaraties in het StyleSheet-object. In het volgende voorbeeld wordt aangegeven hoe een extern CSS-bestand kan worden gelezen en de stijldeclaraties ervan kunnen worden toegepast op een TextField-object.

Ten eerste is hier de inhoud van het CSS-bestand dat moet worden geladen, met de naam 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;
}
```

Vervolgens is er de ActionScript-code voor een klasse waarmee het bestand example.css wordt geladen en de stijlen op de inhoud van het tekstveld worden toegepast:

```
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;
        }
    }
}
```

Wanneer de CSS-gegevens zijn geladen, wordt de methode `onCSSFileLoaded()` uitgevoerd en wordt de methode `StyleSheet.parseCSS()` aangeroepen om de stijldeclaraties over te brengen naar het StyleSheet-object.

Tekstbereiken in een tekstveld opmaken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een nuttige methode van de klasse flash.text.TextField is de methode `setTextFormat()`. Met `setTextFormat()` kunt u specifieke eigenschappen toewijzen aan de inhoud van een deel van een tekstveld, zodat dit reageert op de invoer van de gebruiker. Dit kunnen bijvoorbeeld formulieren zijn waarmee gebruikers eraan worden herinnerd dat bepaalde invoer nodig is of om de nadruk van een subsectie van een tekstgedeelte in een tekstveld te wijzigen wanneer de gebruiker delen van de tekst selecteert.

In het volgende voorbeeld wordt `TextField.setTextFormat()` voor een reeks tekens gebruikt om de weergave van een deel van de inhoud van `myTextField` te wijzigen wanneer de gebruiker op het tekstveld klikt.

```
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);
}
```

Geavanceerde tekstrendering

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 biedt verschillende klassen in het flash.text-pakket voor het sturen van de eigenschappen van weergegeven tekst, waaronder ingesloten lettertypen, instellingen voor antialiasing, alfakanaalbesturing en andere specifieke instellingen. De Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform biedt gedetailleerde beschrijvingen van deze klassen en eigenschappen, inclusief de klassen CSMSettings, Font en TextRenderer.

Ingesloten lettertypen gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een specifiek lettertype opgeeft voor een tekstveld in de toepassing, zoekt Flash Player of AIR naar een apparaatlettertype (een lettertype dat zich op de computer van de gebruiker bevindt) met dezelfde naam. Als het lettertype niet op het systeem wordt aangetroffen, of als de gebruiker een iets andere versie heeft van een lettertype met die naam, kan de tekstweergave sterk afwijken van wat u in gedachte had. Standaard wordt de tekst in een Times Roman-lettertype weergegeven.

Om te zorgen dat de gebruiker precies het juiste lettertype ziet, kunt u dat lettertype insluiten in het SWF-bestand van de toepassing. Ingesloten lettertypen hebben een aantal voordelen:

- Tekens van ingesloten lettertypen zijn antialiased, waardoor de randen ervan er vloeiender uitzien, vooral bij grotere tekst.
- U kunt tekst die gebruik maakt van ingesloten lettertypen roteren.
- Tekst met ingesloten lettertypen kan transparant of semitransparant worden gemaakt.
- U kunt de CSS-stijl `kerning` gebruiken bij ingesloten lettertypen.

De belangrijkste beperking bij het gebruik van ingesloten lettertypen is dat hierdoor de bestandsgrootte of downloadgrootte van de toepassing toeneemt.

Afhankelijk van uw ontwikkelingsomgeving, verschilt de exacte methode waarop een lettertypebestand in het SWF-bestand van uw toepassing wordt ingesloten.

Wanneer u een lettertype hebt ingesloten, kunt u zorgen dat een tekstveld het juiste ingesloten lettertype gebruikt:

- Stel de eigenschap `embedFonts` van het tekstveld in op `true`.
- Maak een TextFormat-object, stel de eigenschap `fontFamily` ervan in op de naam van het ingesloten lettertype en pas het TextFormat-object toe op het tekstveld. Wanneer u een ingesloten lettertype opgeeft, moet de eigenschap `fontFamily` slechts één naam bevatten. Er kan geen door komma's gescheiden lijst van meerdere lettertypenamen worden gebruikt.
- Als u CSS-stijlen gebruikt om lettertypen in te stellen voor TextFields of componenten, stelt u de CSS-eigenschap `font-family` in op de naam van het ingesloten lettertype. De eigenschap `font-family` moet één naam bevatten en geen lijst met namen als u een ingesloten lettertype wilt opgeven.

Een lettertype insluiten in Flash

Met Flash Professional kunt u bijna elk lettertype dat op uw systeem is geïnstalleerd, insluiten, inclusief TrueType-lettertypen en Type 1 Postscript-lettertypen.

U kunt lettertypen op veel manieren in een toepassing insluiten, zoals:

- De lettertype- en stijleigenschappen van een tekstveld instellen in het werkgebied en klikken op het selectievakje Lettertypen insluiten
- Een lettertypesymbool maken en hiernaar verwijzen
- Een tijdens runtime gedeelde bibliotheek met symbolen van ingesloten lettertypen maken en gebruiken

Zie “Ingesloten lettertypen voor dynamische of invoertekstvelden” in *Flash gebruiken* voor meer informatie over het insluiten van lettertypen in toepassingen.

Lettertypen insluiten in Flex

U kunt lettertypen op veel manieren insluiten in een Flex-toepassing, zoals:

- Met de metagegevenstag `[Embed]` in een script
- Met de stijldeclaratie `@font-face`
- Stel een klasse voor het lettertype vast en gebruik de tag `[Embed]` om het lettertype in te sluiten.

U kunt alleen TrueType-lettertypen rechtstreeks insluiten in een Flex-toepassing. Lettertypen in andere indelingen, zoals Type 1 Postscript-lettertypen, kunnen eerst met Flash Professional worden ingesloten in een SWF-bestand dat u vervolgens gebruikt in uw Flex-toepassing. Zie 'Lettertypen van SWF-bestanden insluiten' in *Flex 4 gebruiken*

Meer Help-onderwerpen

Lettertypen insluiten voor een consistent uiterlijk van tekst

Peter deHaan: Embedding fonts

Divillysausages.com: AS3 Font embedding masterclass

Scherpte, dikte en antialiasing regelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Standaard bepaalt Flash Player of AIR de instellingen voor tekstweergave-elementen zoals scherpte, dikte en antialiasing als de grootte van de tekst wordt aangepast, de kleur wordt gewijzigd of de tekst wordt weergegeven op verschillende achtergronden. In sommige gevallen, zoals wanneer u zeer kleine of zeer grote tekst hebt, of tekst tegen een verscheidenheid aan unieke achtergronden, wilt u wellicht de controle houden over deze instellingen. U kunt de instellingen van Flash Player of AIR negeren met de klasse `flash.text.TextRenderer` en de bijbehorende klassen, zoals de klasse CSMSettings. Deze klassen geven u precieze controle over de weergavekwaliteit van ingesloten tekst. Zie “Ingesloten lettertypen gebruiken” op pagina 405 voor meer informatie over ingesloten lettertypen.

Opmerking: *De eigenschap `flash.text.TextField.antiAliasType` moet de waarde `AntiAliasType.ADVANCED` hebben om de scherpte, dikte of de eigenschap gridFitType te kunnen instellen of om de methode `TextRenderer.setAdvancedAntiAliasingTable()` te kunnen gebruiken.*

In het volgende voorbeeld worden aangepaste CSM-eigenschappen (Continuous Stroke Modulation) en opmaak toegepast op weergegeven tekst met behulp van een ingesloten lettertype met de naam `myFont`. Wanneer de gebruiker op de weergegeven tekst klikt, past Flash Player of Adobe AIR de aangepaste instellingen toe:

```
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);
}
```

Werken met statische tekst

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Statische tekst wordt alleen met Flash Professional gemaakt. U kunt statische tekst niet programmatisch instantiëren met behulp van ActionScript. Statische tekst is nuttig als de tekst kort is en niet bedoeld is om te worden gewijzigd (zoals bij dynamische tekst mogelijk is). U moet statische tekst zien als een soort grafisch element, zoals een cirkel of vierkant, dat in het werkgebied in Flash Professional wordt getekend. Statische tekst heeft meer beperkingen dan dynamische tekst, maar ActionScript 3.0 biedt u wel de mogelijkheid om de eigenschapwaarden van statische tekst te lezen met behulp van de klasse StaticText. Met de klasse TextSnapshot kunt u ook waarden uit de statische tekst lezen.

Toegang tot statische tekstvelden met de klasse StaticText

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse flash.text.StaticText in het deelvenster Handelingen van Flash Professional gebruiken om te communiceren met een instantie van statische tekst die in het werkgebied is geplaatst. U kunt ook werken in ActionScript-bestanden die communiceren met een SWF-bestand dat statische tekst bevat. In beide gevallen kunt u geen instantie van statische tekst programmatisch instantiëren. Statische tekst wordt met Flash Professional gemaakt.

Als u een verwijzing naar een bestaand statisch tekstveld wilt maken, doorloopt u de items in de weergegeven lijst en wijst u een variabele toe. Bijvoorbeeld:

```
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);
}
}
```

Wanneer u een referentie naar een statisch tekstveld hebt, kunt u de eigenschappen van dat veld gebruiken in ActionScript 3.0. De volgende code wordt gekoppeld aan een frame in de tijdlijn en gaat ervan uit dat een variabele met de naam `myFieldLabel` is toegewezen aan een referentie naar statische tekst. Een dynamisch tekstveld `myField` wordt ten opzichte van de waarden `x` en `y` geplaatst van `myFieldLabel` en geeft opnieuw de waarde van `myFieldLabel` weer.

```
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
```

De klasse TextSnapshot gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u programmatisch wilt werken met een bestaande instantie van statische tekst, kunt u de klasse flash.text.TextSnapshot gebruiken om met de eigenschap `textSnapshot` van een flash.display.DisplayObjectContainer te werken. Met andere woorden, u maakt een TextSnapshot-instantie van de eigenschap `DisplayObjectContainer.textSnapshot`. U kunt vervolgens methoden toepassen op die instantie om waarden op te halen of delen van de statische tekst te selecteren.

Plaats bijvoorbeeld een statisch tekstveld dat de tekst 'TextSnapshot Example' bevat op het werkgebied. Voeg de volgende ActionScript toe aan frame 1 van de tijdlijn:

```
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);
```

De klasse TextSnapshot is handig om tekst uit statische tekstvelden in een geladen SWF-bestand te halen, als u de tekst wilt gebruiken als een waarde in een ander deel van een toepassing.

TekstField-voorbeeld: tekstopmaak in krantenstijl

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het voorbeeld met nieuwslay-out wordt tekst opgemaakt als een artikel in een gedrukte krant. De invoertekst kan een kop, een subkop en de hoofdtekst van het artikel bevatten. Bij een bepaalde weergavebreedte en -hoogte, worden met dit nieuwslay-outvoorbeeld de kop en subkop zodanig opgemaakt dat de volledige breedte van het weergavegebied wordt gebruikt. De tekst van het artikel wordt verspreid over twee of meer kolommen.

In dit voorbeeld worden de volgende ActionScript-programmeringstechnieken geïllustreerd:

- De klasse TextField uitbreiden
- Een extern CSS-bestand laden en toepassen
- CSS-stijlen converteren naar TextFormat-objecten
- De klasse TextLineMetrics gebruiken om informatie over de tekstweergavegrootte te krijgen

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van News Layout vindt u in de map Samples/NewsLayout. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
NewsLayout.mxml of NewsLayout.fla	De gebruikersinterface voor de toepassing voor Flex (MXML) of Flash (FLA).
com/example/programmingas3/newslayout/StoryLayoutComponent.as	Een Flex-klasse UIComponent die de instantie StoryLayout plaatst.
com/example/programmingas3/newslayout/StoryLayout.as	De hoofdklasse van ActionScript waarmee alle componenten van een nieuwsartikel worden gerangschikt voor weergave.
com/example/programmingas3/newslayout/FormattedTextField.as	Een subklasse van de klasse TextField waarmee het eigen TextFormat-object wordt beheerd.
com/example/programmingas3/newslayout/HeadlineTextField.as	Een subklasse van de klasse FormattedTextField waarmee de lettertypegrootte wordt aangepast aan de gewenste breedte.
com/example/programmingas3/newslayout/MultiColumnTextField.as	Een ActionScript-klasse waarmee tekst over twee of meer kolommen wordt verdeeld.
story.css	Een CSS-bestand dat tekststijlen definieert voor de lay-out.

Het externe CSS-bestand lezen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De nieuwslay-outtoepassing start met het lezen van de artikeltekst uit een lokaal XML-bestand. Vervolgens wordt een extern CSS-bestand gelezen dat de opmaakinformatie bevat voor de kop, subkop en hoofdtekst.

In het CSS-bestand worden drie stijlen gedefinieerd: een standaardalineastijl voor het artikel en de stijlen h1 en h2 voor respectievelijk de kop en de subkop.

```
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;
}
```

De techniek die wordt gebruikt voor het lezen van het externe CSS-bestand is dezelfde als de techniek die wordt beschreven in "Een extern CSS-bestand laden" op pagina 403. Wanneer het CSS-bestand is geladen, voert de toepassing de hieronder getoonde methode `onCSSFileLoaded()` uit.

```
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();
}
```

Met de methode `onCSSFileLoaded()` wordt een StyleSheet-object gemaakt, dat de CSS-invoergegevens parseert. De hoofdtekst van het artikel wordt weergegeven in een MultiColumnTextField-object, dat rechtstreeks gebruik kan maken van een StyleSheet-object. Het kopveld maakt echter gebruik van de klasse HeadlineTextField, die een TextFormat-object gebruikt voor de opmaak.

Met de methode `onCSSFileLoaded()` wordt tweemaal de methode `getTextStyle()` aangeroepen om een CSS-stijldeclaratie te converteren naar een TextFormat-object dat kan worden gebruikt bij elk van de twee HeadlineTextField-objecten.

```
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;
}
```

De namen van de eigenschappen en de betekenis van de eigenschapwaarden verschillen tussen CSS-stijldeclaraties en TextFormat-objecten. Met de methode `getTextStyle()` worden CSS-eigenschapwaarden vertaald in de waarden die worden verwacht door het TextFormat-object.

Artikelelementen op de pagina rangschikken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse StoryLayout worden de kop-, subkop- en hoofdtekstvelden opgemaakt en ingedeeld in een krantenopmaak. Met de methode `displayText()` worden in eerste instantie de verschillende velden gemaakt en geplaatst.

```
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;
...
```

Elk veld wordt onder het vorige veld geplaatst door de eigenschap `y` ervan gelijk te maken aan de eigenschap `y` van het vorige veld plus de hoogte van het veld. Deze dynamische plaatsingsberekening is nodig omdat HeadlineTextField-objecten en MultiColumnTextField-objecten hun hoogte kunnen aanpassen aan de inhoud.

Tekengrootte aanpassen aan de veldgrootte

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij een bepaalde breedte in pixels en een maximumaantal regels dat kan worden weergegeven, past HeadlineTextField de tekengrootte aan zodat de tekst in het veld past. Als de tekst kort is, is de tekengrootte groot waardoor een koptekst in tabloidstijl ontstaat. Als de tekst lang is, wordt de tekengrootte kleiner.

De hieronder getoonde methode `HeadlineTextField.fitText()` past de tekengrootte aan:

```
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;
    }
}
```

De methode `HeadlineTextField.fitText()` maakt gebruik van een eenvoudige recursieve techniek om de tekengrootte aan te passen. Eerst wordt een schatting gemaakt van het gemiddelde aantal pixels per teken in de tekst en op basis daarvan wordt een eerste puntgrootte berekend. Vervolgens wordt de tekengrootte gewijzigd en wordt gecontroleerd of er tekstomloop heeft plaatsgevonden om meer dan één regel tekst te maken. Als er te veel regels zijn, wordt de methode `shrinkText()` aangeroepen waarmee de tekengrootte wordt verkleind en opnieuw wordt gekeken. Als er niet te veel regels zijn, wordt de methode `growText()` aangeroepen waarmee de tekengrootte wordt vergroot en opnieuw wordt gekeken. Het proces stopt op het moment waarop het vergroten van de tekengrootte met één punt te veel regels zou opleveren.

Tekst splitsen over meerdere kolommen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse MultiColumnTextField wordt tekst verspreid over meerdere TextField-objecten die vervolgens worden gerangschikt als krantenkolommen.

Met de constructor `MultiColumnTextField()` wordt eerst een array van TextField-objecten gemaakt, één voor elke kolom, zoals hier aangegeven:

```
    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);
    }
```

Elk TextField-object wordt met de methode `addChild()` toegevoegd aan de array en aan de weergavelijst.

Telkens wanneer de StoryLayout-eigenschap `text` of de eigenschap `styleSheet` wijzigt, wordt de methode `layoutColumns()` aangeroepen om de tekst opnieuw weer te geven. Met de methode `layoutColumns()` wordt de methode `getOptimalHeight()` aangeroepen om de juiste pixelhoogte te berekenen die nodig is om alle tekst binnen de beschikbare lay-outbreedte te passen.

```
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;
    }
}
```

Met de methode `getOptimalHeight()` wordt eerst de breedte van elke kolom berekend. Vervolgens worden de breedte en de eigenschap `htmlText` van het eerste TextField-object in de array ingesteld. De methode `getOptimalHeight()` gebruikt dat eerste TextField-object om het totale aantal regels met regelomloop in de tekst te berekenen, en bepaalt op basis daarvan hoeveel regels elke kolom moet hebben. Vervolgens wordt de methode `TextField.getLineMetrics()` aangeroepen om een TextLineMetrics-object op te halen dat gegevens bevat over de grootte van de tekst op de eerste regel. De eigenschap `TextLineMetrics.height` geeft de volledige hoogte van een regel tekst weer (in pixels), inclusief stok, staart en regelafstand. De optimale hoogte voor het MultiColumnTextField-object is vervolgens de regelhoogte maal het aantal regels per kolom, plus 4 voor de rand van twee pixels boven en onder aan een TextField-object.

Hier is de code voor de volledige methode `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);
        }
    }
    }
}
```

Wanneer de eigenschap `preferredHeight` is ingesteld door het aanroepen van de methode `getOptimalHeight()`, worden met de methode `layoutColumns()` de TextField-objecten doorlopen en wordt de hoogte van elk van de objecten ingesteld op de waarde `preferredHeight`. Met de methode `layoutColumns()` worden vervolgens net voldoende regels tekst in elk veld verdeeld zodat er geen verschuiving optreedt in de afzonderlijke velden en de tekst in elk volgende veld begint waar de tekst in het vorige veld eindigde. Als de tekstuitlijningsstijl is ingesteld op "uitvullen", wordt de methode `justifyLastLine()` aangeroepen om de laatste regel tekst in een veld uit te vullen. Anders zou de laatste regel worden behandeld als een regel aan het einde van een alinea en niet worden uitgevuld.

Hoofdstuk 22: De Flash Text Engine gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De Adobe® Flash® Text Engine, die beschikbaar is vanaf Flash Player 10 en Adobe® AIR™ 1.5, biedt ondersteuning op een laag niveau voor geavanceerde besturing van tekstmeeteenheden, opmaak en bidirectionele tekst. De Flash Text Engine biedt verbeterde tekstdoorloop en uitgebreide taalondersteuning. Hoewel de FTE kan worden gebruikt om eenvoudige tekstelementen te maken en te beheren, is de FTE hoofdzakelijk ontworpen als basis voor ontwikkelaars om componenten voor het afhandelen van tekst te maken. Als zodanig is voor de Flash Text Engine een hoger niveau van programmeringskennis nodig. Zie "De klasse TextField gebruiken" op pagina 393 om eenvoudige tekstelementen weer te geven.

Het Text Layout Framework bevat een tekstafhandelingscomponent op basis van de FTE en biedt een eenvoudigere manier om de geavanceerde functies ervan te gebruiken. Het Text Layout Framework is een uitbreidbare bibliotheek die geheel in ActionScript 3.0 is gemaakt. U kunt gebruik maken van de bestaande TLF-component of van het framework om uw eigen tekstcomponent te maken. Zie "Text Layout Framework gebruiken" op pagina 448 voor meer informatie.

Meer Help-onderwerpen

flash.text.engine-pakket

Tekst maken en weergeven

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de klassen waaruit de Flash Text Engine bestaat, kunt u tekst maken, opmaken en besturen. De volgende klassen vormen de elementaire bouwstenen voor het maken en weergeven van tekst met de Flash Text Engine:

- TextElement/GraphicElement/GroupElement: deze bevatten de inhoud van een TextBlock-instantie
- ElementFormat: hiermee worden opmaakkenmerken aangegeven voor de inhoud van een TextBlock-instantie
- TextBlock: de plaats waar een alinea tekst wordt opgebouwd
- TextLine: een regel tekst die is gemaakt van het TextBlock

Als u tekst wilt weergeven, maakt u een TextElement-object van een String met behulp van een ElementFormat-object om de opmaakkenmerken te definiëren. Vervolgens wijst u het TextElement-object toe aan de eigenschap `content` van een TextBlock-object. Maak de regels tekst die moeten worden weergegeven door de methode `TextBlock.createTextLine()` aan te roepen. De methode `createTextLine()` retourneert een TextLine-object dat het deel van de tekenreeks bevat dat in de opgegeven breedte past. Roep de methode herhaaldelijk aan totdat de volledige tekenreeks in de regels past. Wanneer er geen regels meer moeten worden gemaakt, wordt de waarde `TextLineCreationResult.COMPLETE` toegewezen aan de eigenschap textLineCreationResult van het TextBlock-object. Als u de regels wilt weergeven, voegt u deze toe aan de weergave lijst (met bijbehorende positiewaarden `x` en `y`).

In de volgende code worden bijvoorbeeld deze FTE-klassen gebruikt om 'Hello World! This is Flash Text Engine!" weer te geven, met standaardopmaak en -lettertypewaarden. In dit eenvoudige voorbeeld wordt slechts één tekstregel gemaakt.

```
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;
        }
    }
}
```

De parameters voor `createTextLine()` geven de regel op waarvan de nieuwe regel moet beginnen en de breedte van de regel in pixels. De regel waarvan de nieuwe regel moet beginnen is doorgaans de vorige regel, maar in het geval van de eerste regel is dit `null`.

GraphicElement- en GroupElement-objecten toevoegen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een GraphicElement-object toewijzen aan een TextBlock-object om een afbeelding of een grafisch element weer te geven. Maak eenvoudigweg een instantie van de klasse GraphicElement vanuit een afbeelding en wijs de instantie toe aan de eigenschap `TextBlock.content`. Maak de tekstregel door `TextBlock.createTextline()` aan te roepen zoals u normaal zou doen. In het volgende voorbeeld worden twee tekstregels gemaakt: één met een GraphicElement-object en één met een TextElement-object.

```
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;
        }
    }
}
```

U kunt een GroupElement-object maken om een groep te maken van TextElement-, GraphicElement- en andere GroupElement-objecten. U kunt een GroupElement toewijzen aan de eigenschap `content` van een TextBlock-object. De parameter voor de constructor `GroupElement()` is een vector, die wijst naar de tekst, afbeelding en groepselementen die de groep uitmaken. In het volgende voorbeeld worden twee grafische elementen en een tekstelement gegroepeerd en worden deze als eenheid toegewezen aan een tekstblok.

```
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;
        }
    }
}
```

Tekst vervangen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt tekst in een TextBlock-instantie vervangen door `TextElement.replaceText()` aan te roepen om tekst te vervangen in het TextElement dat u hebt toegewezen aan de eigenschap `TextBlock.content`.

In het volgende voorbeeld wordt `replaceText()` gebruikt om tekst aan het begin van de regel in te voegen, tekst aan het einde van de regel toe te voegen en tekst in het midden van de regel te vervangen.

```
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);
        }
    }
}
```

Met de methode `replaceText()` vervangt u tekst die is opgegeven in de parameters `beginIndex` en `endIndex` door de tekst die is aangegeven door de paramater `newText`. Als de waarden van de parameters `beginIndex` en `endIndex` hetzelfde zijn, voegt `replaceText()` de opgegeven tekst in op die locatie. Anders worden de tekens die worden aangegeven met `beginIndex` en `endIndex` vervangen door de nieuwe tekst.

Gebeurtenissen afhandelen in FTE

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt gebeurtenislisteners toevoegen aan een TextLine-instantie, net zoals u dit kunt bij andere weergaveobjecten. U kunt bijvoorbeeld detecteren wanneer een gebruiker met de muis over een tekstregel beweegt of op de regel klikt. In het volgende voorbeeld worden beide gebeurtenissen gedetecteerd. Wanneer u de muis over de regel beweegt, verandert de cursor in een knopcursor en wanneer u op de regel klikt, verandert deze van kleur.

```
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);
        }
    }
}
```

Gebeurtenissen spiegelen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt ook gebeurtenissen op een tekstblok, of op een deel van een tekstblok, spiegelen naar een gebeurtenisdispatcher. Maak eerste een EventDispatcher-instantie en wijs deze vervolgens toe aan de eigenschap `eventMirror` van een TextElement-instantie. Als het tekstblok bestaat uit één tekstelement, spiegels de text engine gebeurtenissen voor het hele tekstblok. Als het tekstblok bestaat uit meerdere tekstelementen, spiegelt de text engine alleen gebeurtenissen voor de TextElement-instantie waarvan de eigenschap `eventMirror` is ingesteld. De tekst in het volgende voorbeeld bestaat uit drie elementen: het woord "Click", het woord "here" en de tekenreeks "to see me in italic". In het voorbeeld wordt een gebeurtenisdispatcher toegewezen aan het tweede tekstelement, het woord "here" en wordt een gebeurtenislistener, de methode `clickHandler()`, toegevoegd. Met de methode `clickHandler()` wordt de tekst cursief gemaakt. Ook wordt de inhoud van het derde tekstelement vervangen zodat er komt te staan: "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);
        }
    }
}
```

Met de functies `mouseOverHandler()` en `mouseOutHandler()` verandert de cursor in een knopcursor wanneer deze zich boven het woord "here" bevindt en weer in een pijl als dit niet het geval is.

Tekst opmaken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Een `TextBlock`-object is plaats waar regels tekst worden gemaakt. De inhoud van een `TextBlock` wordt toegewezen via het `TextElement`-object. Een `ElementFormat`-object handelt de opmaak van de tekst af. De klasse ElementFormat definieert eigenschappen zoals basislijnuitlijning, tekenspatiëring, tekstspatiëring, tekstrotatie en tekengrootte, -kleur en hoofd-/kleine letter. De klasse omvat ook een `FontDescription`, die nader wordt beschreven in "Werken met lettertypen" op pagina 430.

Het ElementFormat-object gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De constructor voor het `ElementFormat`-object kan worden gebruikt met een lange lijst optionele parameters, waaronder een `FontDescription`. U kunt deze eigenschappen ook buiten de constructor instellen. In het volgende voorbeeld wordt de relatie getoond van de verschillende objecten bij het definiëren en weergeven van een eenvoudige tekstregel:

```
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);
        }
    }
}
```

Lettertypekleur en transparantie (alfa)

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de eigenschap `color` van het `ElementFormat`-object wordt de lettertypekleur ingesteld. De waarde is een geheel getal dat de RGB-componenten van de kleur weergeeft; bijvoorbeeld 0xFF0000 voor rood en 0x00FF00 voor groen. De standaardwaarde is zwart (0x000000).

Met de eigenschap `alpha` wordt de alfatransparantiewaarde van een element ingesteld (zowel `TextElement` als `GraphicElement`). Waarden kunnen variëren van 0 (volledig transparant) tot 1 (volledig ondoorzichtig, hetgeen de standaardwaarde is). Elementen met een `alpha` van 0 zijn onzichtbaar, maar wel actief. Deze waarde wordt vermenigvuldigd met eventuele overgeërfde alfawaarden, waardoor het element transparanter wordt.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

Basislijnuitlijning en -verschuiving

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het lettertype en de grootte van de grootste tekst op een regel bepaalt de dominante basislijn. U kunt deze waarde overschrijven door `TextBlock.baselineFontDescription` en `TextBlock.baselineFontSize` in te stellen. U kunt de dominante basislijn uitlijnen met een van een reeks basislijnen in de tekst. Deze basislijnen bevatten de stoklijn en de staartlijn of de ideografische boven- of onderkant of het ideografische midden.

***A.** Stok **B.** Basislijn **C.** Staart **D.** x-hoogte*

In het `ElementFormat`-object bepalen drie eigenschappen de basislijn- en uitlijningskenmerken. Met de eigenschap `alignmentBaseline` wordt de hoofdbasislijn van een `TextElement` of `GraphicElement` ingesteld. Deze basislijn is de "magnetische" lijn voor het element en tevens de positie waarop de dominante basislijn van alle tekst wordt uitgelijnd.

Met de eigenschap `dominantBaseline` wordt bepaald welke van de verschillende basislijnen van het element moet worden gebruikt. Dit bepaalt de verticale positie van het element op de lijn. De standaardwaarde is `TextBaseline.ROMAN`, maar dit kan ook zodanig worden ingesteld dat de `IDEOGRAPHIC_TOP`- of `IDEOGRAPHIC_BOTTOM`-basislijn dominant is.

Met de eigenschap `baselineShift` wordt de basislijn een aantal pixels op de y-as verplaatst. Bij normale (niet geroteerde) tekst wordt de basislijn met een positieve waarde omlaag en met een negatieve waarde omhoog verplaatst.

Hoofdletters/kleine letters

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de eigenschap `TypographicCase` van `ElementFormat` wordt aangegeven of tekst in hoofdletters, kleine letters of als kleinkapitalen moet worden weergegeven.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

Tekst roteren

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

U kunt een blok tekst of de glyphs in een tekstsegment in stappen van 90 graden roteren. Met de klasse TextRotation worden de volgende constanten gedefinieerd voor het instellen van zowel tekstblok- als glyphrotatie:

Constante	Waarde	Beschrijving
AUTO	"auto"	Geeft een rotatie van 90 graden linksom aan. Dit wordt doorgaans gebruikt bij verticale Aziatische tekst om alleen glyphs te roteren die rotatie behoeven.
ROTATE_0	"rotate_0"	Geeft geen rotatie aan.
ROTATE_180	"rotate_180"	Geeft 180 graden rotatie aan
ROTATE_270	"rotate_270"	Geeft 270 graden rotatie aan.
ROTATE_90	"rotate_90"	Geeft een rotatie van 90 rechtsom aan.

Als u de regels tekst in een tekstblok wilt roteren, stelt u de eigenschap `TextBlock.lineRotation` in voordat u de methode `TextBlock.createTextLine()` aanroept om de tekstregel te maken.

Als u de glyphs in een blok tekst of een segment wilt roteren, stelt u de eigenschap `ElementFormat.textRotation` in op het aantal graden dat u de glyphs wilt roteren. Een glyph is de vorm waaruit een teken bestaat of een deel van een teken dat uit meerdere glyphs bestaat. De letter a en de punt op een i zijn bijvoorbeeld glyphs.

Het roteren van glyphs is relevant in sommige Aziatische talen, waarin u de regels naar verticaal wilt roteren, maar niet de tekens binnen de regels. Zie "Oost-Aziatische tekst uitvullen" op pagina 434 voor meer informatie over het roteren van Aziatische tekst.

Hier is een voorbeeld van het roteren van zowel een blok tekst als de glyphs daarin, zoals gebruikelijk bij Aziatische tekst. In het voorbeeld wordt ook een Japans lettertype gebruikt:

```
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);
        }
    }
}
```

ElementFormat vergrendelen en klonen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Wanneer een `ElementFormat`-object wordt toegewezen aan een type `ContentElement`, wordt de eigenschap `locked` automatisch ingesteld op `true`. Als geprobeerd wordt een vergrendeld `ElementFormat`-object te wijzigen, leidt dit tot een `IllegalOperationError`. De beste manier is om een dergelijk object volledig te definiëren voordat u het toewijst aan een `TextElement`-instantie.

Als u een bestaande `ElementFormat`-instantie wilt wijzigen, controleert u eerst de eigenschap `locked` ervan. Als deze is ingesteld op `true`, gebruikt u de methode `clone()` om een ontgrendelde kopie van het object te maken. De eigenschappen van dit onvergrendelde object kunnen worden gewijzigd en dit object kan vervolgens worden toegewezen aan de `TextElement`-instantie. Nieuwe regels die hiermee worden gemaakt, hebben de nieuwe opmaak. Eerdere regels die zijn gemaakt met hetzelfde object en die de oude opmaak gebruiken, blijven ongewijzigd.

```
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);
        }
    }
}
```

Werken met lettertypen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het `FontDescription`-object wordt gebruikt in combinatie met `ElementFormat` om een lettertype te identificeren en enkele van de kenmerken ervan te definiëren. Deze kenmerken zijn onder andere de lettertypenaam, dikte, stand, en waar het lettertype kan worden gevonden (op het apparaat of ingesloten).

Opmerking: *FTE biedt geen ondersteuning voor Type 1-lettertypen of bitmaplettertypen zoals Type 3, ATC, sfnt-wrapped CID of Naked CID.*

Lettertype-eigenschappen definiëren (FontDescription-object)

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De eigenschap `fontName` van het `FontDescription`-object kan één naam zijn of een door komma's gescheiden lijst namen. In een lijst zoals 'Arial, Helvetica, _sans' zoekt de text engine eerst naar 'Arial', vervolgens naar 'Helvetica' en ten slotte naar '_sans' als het programma de eerste twee lettertypen niet kan vinden. De set lettertypenamen omvat drie generieke apparaatlettertypenamen: "_sans", "_serif" en "_typewriter". Ze verwijzen ook naar specifieke apparaatlettertypen, afhankelijk van het afspeelsysteem. Het is goed gebruik om standaardnamen zoals deze te specificeren in alle lettertypebeschrijvingen die apparaatlettertypen gebruiken. Als er geen `fontName` is opgegeven, wordt standaard "_serif" gebruikt.

De eigenschap `fontPosture` kan worden ingesteld op de standaardwaarde (`FontPosture.NORMAL`) of op cursief (`FontPosture.ITALIC`). De eigenschap `fontWeight` kan worden ingesteld op de standaardwaarde (`FontWeight.NORMAL`) of op vet (`FontWeight.BOLD`).

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

Ingesloten versus apparaatlettertypen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De eigenschap `fontLookup` van het `FontDescription`-object geeft aan of de text engine naar een apparaatlettertype of naar een ingesloten lettertype zoekt om tekst te renderen. Als er een apparaatlettertype (`FontLookup.DEVICE`) is gespecificeerd, zoekt de runtime op het afspeelsysteem naar het lettertype. Als een ingesloten lettertype is opgegeven (`FontLookup.EMBEDDED_CFF`), zoekt de runtime naar een ingesloten lettertype met de opgegeven naam in het SWF-bestand. Alleen ingesloten CFF-lettertypen (Compact Font Format) werken met deze instelling. Als het opgegeven lettertype niet wordt gevonden, wordt teruggevallen op een apparaatlettertype.

Apparaatlettertypen leveren een kleiner SWF-bestand op. Ingesloten lettertypen geven een meer getrouwe weergave op verschillende platforms.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

Rendermodus en aanwijzingen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

CFF-rendering (Compact Font Format) is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5. Dit type rendering van lettertypen maakt tekst leesbaarder en zorgt voor een hogere kwaliteit weergave van lettertypen bij kleine tekengroottes. Deze instelling geldt alleen voor ingesloten lettertypen. `FontDescription` staat standaard op deze instelling (`RenderingMode.CFF`) voor de eigenschap `renderingMode`. U kunt deze eigenschap instellen op `RenderingMode.NORMAL` zodat het type rendering overeenkomt met het type dat wordt gebruikt door Flash Player 7 of eerdere versies.

Wanneer CFF-rendering is geselecteerd, regelt een tweede eigenschap `cffHinting`, hoe de horizontale stammen van een lettertype worden afgezet op het subpixelraster. De standaardwaarde, `CFFHinting.HORIZONTAL_STEM`, maakt gebruik van CFF-aanwijzingen. Als deze eigenschap wordt ingesteld op `CFFHinting.NONE`, worden de aanwijzingen verwijderd, hetgeen toepasselijk is bij animatie of grotere tekengroottes.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

FontDescription vergrendelen en klonen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Wanneer een `FontDescription`-object wordt toegewezen aan een `ElementFormat`, wordt de eigenschap `locked` ervan automatisch ingesteld op `true`. Als geprobeerd wordt een vergrendeld `FontDescription`-object te wijzigen, leidt dit tot een `IllegalOperationError`. De beste manier is om een dergelijk object volledig te definiëren voordat u het toewijst aan een `ElementFormat`.

Als u een bestaande `FontDescription` wilt wijzigen, controleert u eerst de eigenschap `locked` ervan. Als deze is ingesteld op `true`, gebruikt u de methode `clone()` om een onvergrendelde kopie van het object te maken. De eigenschappen van dit onvergrendelde object kunnen worden gewijzigd en dit object kan vervolgens worden toegewezen aan de `ElementFormat`. Nieuwe regels die worden gemaakt van dit `TextElement` hebben de nieuwe opmaak. Eerdere regels die zijn gemaakt met hetzelfde object, blijven ongewijzigd.

```
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);
        }
    }
}
```

Tekst regelen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

FTE biedt u een nieuwe set tekstopmaakbesturingselementen, waarmee u uitvulling en spatiëring (teken- en tekstspatiëring) kunt afhandelen. Er zijn ook eigenschappen voor het regelen van de manier waarop regels worden afgebroken en voor het instellen van tabstops in regels.

Tekst uitvullen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Door tekst uit te vullen, worden alle regels in een alinea even lang. Dit gebeurt door de tussenruimte tussen woorden en soms tussen letters aan te passen. Het effect is om de tekst aan beide zijden uit te lijnen, terwijl de tussenruimte tussen woorden en letters varieert. Kolommen tekst in kranten en tijdschriften zijn vaak uitgevuld.

Met de eigenschap `lineJustfication` in de klasse SpaceJustifier kunt u de uitvulling van regels in een blok tekst regelen. In de klasse LineJustification worden constanten gedefinieerd die u kunt gebruiken om een uitvuloptie te specificeren: met `ALL_BUT_LAST` worden alle tekstregels behalve de laatste uitgevuld; met `ALL_INCLUDING_LAST` wordt alle tekst, inclusief de laatste regel, uitgevuld; met `UNJUSTIFIED`, de standaardinstelling, wordt alle tekst onuitgevuld gelaten.

Stel voor het uitvullen van tekst de eigenschap `lineJustification` in op een instantie van de klasse SpaceJustifier en wijs die instantie toe aan de eigenschap `textJustifier` van een TextBlock-instantie. In het volgende voorbeeld wordt een alinea gemaakt waarin alle tekst behalve de laatste regel wordt uitgevuld.

```
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);
            }
        }
    }
}
```

Als u de spatiëring niet alleen tussen woorden maar ook tussen letters wilt variëren, stelt u de eigenschap `SpaceJustifier.letterspacing` in op `true`. Door letterspatiëring in te schakelen kunt u lelijke gaten tussen woorden voorkomen, die soms optreedt bij eenvoudige uitvulling.

Oost-Aziatische tekst uitvullen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Bij het uitvullen van Oost-Aziatische tekst gelden enkele overwegingen. Dergelijke tekst kan van boven naar beneden worden geschreven en sommige tekens (de zogenaamde kinsoku), kunnen niet aan het begin of eind van een regel komen. In de klasse JustificationStyle worden de volgende constanten gedefinieerd, waarmee de opties worden aangegeven voor de behandeling van deze tekens. Bij `PRIORITIZE_LEAST_ADJUSTMENT` wordt de uitvulling gebaseerd op het uitrekken of samendrukken van de regel, afhankelijk van wat het beste resultaat oplevert. Bij `PUSH_IN_KINSOKU` wordt de uitvulling gebaseerd op het samendrukken van de tekst als er aan het einde van de regel een kinsoku-teken staat, of het uitrekken als er geen kinsoku is of als er onvoldoende ruimte is.

Bij PUSH_OUT_ONLY wordt de uitvulling gebaseerd op het uitrekken van de regel. Als u een blok verticale Aziatische tekst wilt maken, stelt u de eigenschap TextBlock.lineRotation in op TextRotation.ROTATE_90 en stelt u de eigenschap ElementFormat.textRotation in op TextRotation.AUTO (de standaardinstelling). Door de eigenschap textRotation in te stellen op AUTO zorgt u dat de glyphs in de tekst verticaal blijven in plaats van op hun kant draaien wanneer de regel wordt geroteerd. Met de instelling AUTO worden alleen glyphs van volledige breedte en brede glyphs 90 graden linksom geroteerd, zoals wordt bepaald door de Unicode-eigenschappen van de glyph. In het volgende voorbeeld wordt een verticaal blok Japanse tekst weergegeven en uitgevuld met de optie 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
```

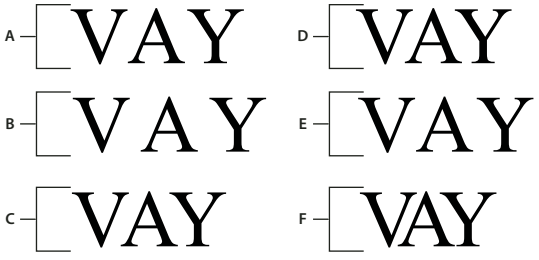
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            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;
            }
        }
    }
}
```

Tekenspatiëring en tekstspatiëring

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Tekenspatiëring en tekstspatiëring hebben invloed op de afstand tussen naast elkaar gelegen tekenparen in een tekstblok. Tekenspatiëring regelt hoe tekenparen zich aan elkaar aanpassen, zoals de paren "WA" of "Va". Tekenspatiëring wordt ingesteld in het `ElementFormat`-object. Dit is standaard ingeschakeld (`Kerning.ON`) en kan worden ingesteld op OFF of AUTO, in welk geval tekenspatiëring alleen wordt toegepast tussen tekens als dit geen Kanji, Hiragana of Katakana is.

Bij tekstspatiëring wordt een bepaald aantal pixels toegevoegd of verwijderd tussen alle tekens in een tekstblok. Dit wordt ook ingesteld in het `ElementFormat`-object. Het werkt zowel bij ingesloten als bij apparaatlettertypen. FTE ondersteunt twee tekstspatiëringseigenschappen, `trackingLeft`, waarbij pixels aan de linkerkant van een teken worden toegevoegd/verwijderd, en `trackingRight`, waarbij pixels aan de rechterkant worden toegevoegd/verwijderd. Als tekenspatiëring wordt gebruikt, wordt de tekstspatiëringswaarde opgeteld bij of afgetrokken van de tekstspatiëringswaarden voor elk tekenpaar.



***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;
```

Regeleinden voor omlooptekst

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De eigenschap `breakOpportunity` van het `ElementFormat`-object bepaalt welke tekens kunnen worden gebruikt voor regeleinden wanneer tekst die omloopt wordt verdeeld in meerdere regels. De standaardinstelling, `BreakOpportunity.AUTO`, gebruikt standaard Unicode-eigenschappen, zoals regeleinden tussen woorden en bij afbreekstreepjes. Wanneer `BreakOpportunity.ALL` wordt gebruikt, kan elk teken worden behandeld als een mogelijkheid voor een regeleinde. Dit is handig voor het creëren van effecten als tekst langs een pad.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

Tabstops

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als u tabstops wilt instellen in een tekstblok, definieert u de tabstops door instanties te maken van de klasse TabStop. De parameters voor de constructor `TabStop()` geven aan hoe de tekst wordt uitgelijnd met de tabstop. Deze parameters geven de positie van de tabstop aan en voor decimale uitlijning de waarde waarop moet worden uitgelijnd, uitgedrukt als een tekenreeks. Deze waarde is standaard een decimaalteken, maar kan ook een komma, een dollarteken of bijvoorbeeld het symbool voor de yen of euro zijn. Met de volgende regel code wordt een tabstop met de naam tab1 gemaakt.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

Wanneer u de tabstops voor een tekstblok hebt gemaakt, wijst u deze toe aan de eigenschap `tabStops` van een TextBlock-instantie. Omdat de eigenschap `tabStops` echter een vector vereist, maakt u eerst een vector en voegt u vervolgens de tabstops toe. Met behulp van de vector kunt u een set tabstops toewijzen aan het tekstblok. In het volgende voorbeeld wordt een `Vector<TabStop>`-instantie gemaakt en wordt hieraan een set TabStop-objecten toegevoegd. Vervolgens worden de tabstops toegewezen aan de eigenschap `tabStops` van een TextBlock-instantie.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Zie "Werken met arrays" op pagina 26 voor meer informatie over vectoren.

In het volgende voorbeeld wordt het effect getoond van elk van de uitlijnopties voor tabstops.

```
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;
            }
        }
    }
}
```

Voorbeeld van de Flash Text Engine: nieuwslay-out

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In dit programmeringsvoorbeeld wordt getoond hoe met de Flash Text Engine een eenvoudige krantenpagina wordt opgemaakt. De pagina bevat een grote kop, een subkop en hoofdtekst die uit meerdere kolommen bestaat.

Eerst maakt u een nieuw FLA-bestand en koppelt u de volgende code aan frame 2 van de standaardlaag.

```
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();
```

StoryLayout.as is het besturingsscript voor dit voorbeeld. In dit script wordt de inhoud ingesteld, de stijlgegevens uit een extern opmaakmodel gelezen en deze stijlen toegewezen aan de ElementFormat-objecten. Daarna worden de tekstelementen voor de kop, de subkop en de diverse kolommen gemaakt.

```
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 wordt gebruikt als basisklasse voor het maken van blokken tekst. Het omvat ook hulpfuncties voor het wijzigen van tekengrootte en hoofdletters/kleine letters.

```
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;
        }
    }
}
```

Met HeadlineTextBlock.as wordt de klasse FormattedTextBlock uitgebreid met een functie voor het maken van titels. Dit omvat een functie voor het passend maken van tekst binnen een afgebakende ruimte op de pagina.

```
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 handelt de opmaak af van tekst binnen een ontwerp met meerdere kolommen. Het toont aan hoe een TextBlock-object op een flexibele manier kan worden gebruikt voor het maken, opmaken en plaatsen van tekstregels.

```
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);
        }

    }
}
```

Hoofdstuk 23: Text Layout Framework gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Overzicht van het Text Layout Framework

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het TLF (Text Layout Framework) is een uitbreidbare ActionScript-bibliotheek. Het TLF is gebaseerd op de text engine in Adobe® Flash® Player 10 en Adobe® AIR® 1.5. Het TLF verschaft geavanceerde functies voor typografie en tekstlay-out voor innovatieve typografie op het web. U kunt het framework gebruiken met Adobe® Flex® of Adobe® Flash® Professional. Ontwikkelaars kunnen bestaande componenten gebruiken of uitbreiden, maar kunnen het framework ook gebruiken om hun eigen tekstcomponenten te maken.

Het TLF biedt de volgende mogelijkheden:

- Bidirectionele tekst, verticale tekst en meer dan 30 scripttalen, waaronder Arabisch, Hebreeuws, Chinees, Japans, Koreaans, Thais, Lao, Vietnamees, enz.
- Het selecteren en bewerken van tekst en het laten doorlopen van tekst over meerdere kolommen en gekoppelde containers
- Verticale tekst, tate-chu-yoko (horizontale tekst binnen verticale tekst) en uitvulling voor Oost-Aziatische typografie
- Veelzijdige typografische controlemiddelen, waaronder tekenspatiëring, ligaturen, typografisch hoofdlettergebruik, cijfertype, cijferbreedte en zachte afbreekstreepjes
- Knippen, kopiëren, plakken, ongedaan maken en standaardbewerkingen met toetsenbord of muis
- Veelzijdige ontwikkelaar-API's voor het bewerken van tekstinhoud, lay-out en opmaak, en het maken van aangepaste tekstcomponenten
- Uitgebreide ondersteuning voor lijsten, waaronder aangepaste markeringen en nummeringsindelingen
- Inlineafbeeldingen en plaatsingsregels

Het TLF is een ActionScript 3.0-bibliotheek die is gemaakt op basis van de text engine van Flash (FTE) die is geïntroduceerd in Flash Player 10. FTE is toegankelijk via het `flash.text.engine`-pakket, dat deel uitmaakt van de Flash Player 10-API (Application Programming Interface).

De Flash Player-API echter biedt eenvoudige toegang tot de text engine. Dit betekent dat sommige taken een relatief groot aantal code vereisen. Het TLF sluit de code van laag niveau in in eenvoudigere API's. Het verschaft ook een conceptuele architectuur die de standaardbouwstenen die door FTE zijn gedefinieerd, indeelt in een gebruiksvriendelijker systeem.

Het TLF is niet zoals FTE geïntegreerd in Flash Player. Het is een onafhankelijke componentenbibliotheek die volledig in ActionScript 3.0 is geschreven. Het framework is uitbreidbaar en kan dus aan specifieke omgevingen worden aangepast. Zowel Flash Professional als de Flex SDK bevatten componenten die zijn gebaseerd op het TLF-framework.

Meer Help-onderwerpen

"Flow"-toepassing met TLF-markeringen

Ondersteuning voor complexe schriften

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het TLF biedt ondersteuning voor complexe schriften. Dit betekent dat schriften die van rechts naar links worden gelezen, kunnen worden weergegeven en bewerkt. Met het TLF kunt u ook een mengeling van talen die van links naar rechts en van rechts naar links worden weergegeven, zoals Arabisch en Hebreeuws, weergeven en bewerken. Het framework ondersteunt niet alleen verticale tekstlay-out voor Chinees, Japans en Koreaans, maar ook tate-chu-yoko (TCY-elementen). TCY-elementen zijn blokken horizontale tekst die in verticale tekstregels zijn ingesloten. De volgende schriften worden ondersteund:

- Latijn (Engels, Spaans, Frans, Vietnamees enzovoort)
- Grieks, Cyrillisch, Armeens, Georgisch en Ethiopisch
- Arabisch en Hebreeuws
- Han-ideografie en Kana (Chinees, Japans en Koreaans) en Hangul Johab (Koreaans)
- Thai, Lao en Khmer
- Devanagari, Bengali, Gurmukhi, Malayalam, Telugu, Tamil, Gujarati, Oriya, Kannada en Tibetaans
- Tifinagh, Yi, Cherokee, Canadese syllabics, Deseret, Shavian, Vai, Tagalog, Hanunoo, Buhid en Tagbanwa

TLF gebruiken in Flash Professional en Flex

U kunt met de TLF-klassen rechtstreeks aangepaste componenten maken in Flash. Bovendien biedt Flash Professional CS5 een nieuwe klasse, namelijk fl.text.TLFTextField, die de TLF-functionaliteit omvat. Gebruik de TLFTextField-klasse om in ActionScript tekstvelden te maken die de geavanceerde tekstweergavefuncties van het TLF gebruiken. Maak een TLFTextField-object op dezelfde manier als wanneer u met de TextField-klasse een tekstveld maakt. Gebruik de `textFlow`-eigenschap om geavanceerde opmaak van de TLF-klassen toe te wijzen.

U kunt Flash Professional ook gebruiken om in het werkgebied met het tekstgereedschap de TLFTextField-instantie te maken. Daarna kunt u met ActionScript de opmaak en lay-out van de inhoud van het tekstveld besturen aan de hand van de TLF-klassen. Zie TLFTextField in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Gebruik de TLF-klassen als u in Flex werkt. Zie “Text Layout Framework gebruiken” op pagina 449 voor meer informatie.

Text Layout Framework gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Gebruik de TFL-klassen als u in Flex werkt of aangepaste tekstcomponenten samenstelt. Het TLF is een ActionScript 3.0-bibliotheek die volledig is opgenomen in de textLayout.swc-bibliotheek. De TLF-bibliotheek bevat ongeveer 100 ActionScript 3.0-klassen en -interfaces die in tien pakketten zijn georganiseerd. Deze pakketten zijn subpakketten van het flashx.textLayout-pakket.

De Text Layout Framework-klassen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De TLF-klassen kunnen in drie categorieën worden gegroepeerd:

- Gegevensstructuren en opmaakklassen
- Renderingklassen
- Gebruikersinteractieklassen

Gegevensstructuren en opmaakklassen

De volgende pakketten bevatten de gegevensstructuren en opmaakklassen voor het TLF:

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

De belangrijkste gegevensstructuur van het TLF is de tekstflowhiërarchie die is gedefinieerd in het elementenpakket. Binnen deze structuur kunt u met het opmaakpakket stijlen en kenmerken toewijzen aan stukken tekst. Met het conversiepakket kunt u bepalen hoe tekst wordt geïmporteerd in en geëxporteerd uit de gegevensstructuur.

Renderingklassen

De volgende pakketten bevatten de renderingklassen voor het TLF:

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

De klassen in deze pakketten maken de rendering mogelijk van tekst die wordt weergegeven door Flash Player. Het fabriekspakket biedt een eenvoudige manier voor het weergeven van statische tekst. Het containerpakket omvat klassen en interfaces waarmee weergavecontainers voor dynamische tekst worden gedefinieerd. Het samenstellingspakket definieert technieken voor het plaatsen en weergeven van dynamische tekst in containers.

Gebruikersinteractieklassen

De volgende pakketten bevatten de Gebruikersinteractieklassen voor het TLF:

- flashx.textLayout.edit
- flashx.textLayout.conversion
- flashx.textLayout

Met de bewerk- en bewerkingen pakketten worden klassen gedefinieerd waarmee u tekst die in de gegevensstructuren is opgeslagen, kunt bewerken. Het gebeurtenis pakket bevat gebeurtenislistener.

Algemene stappen voor het maken van tekst in het Text Layout Framework

Het proces voor het maken van tekst met TLF bestaat uit de volgende stappen:

1. Importeer geformatteerde tekst in de. Zie “Tekst structureren met TLF” op pagina 454 en “Tekst opmaken met TLF” op pagina 458 voor meer informatie.
2. Maak een of meer gekoppelde weergaveobject containers voor de tekst. Zie “Tekstcontainers beheren met TLF” op pagina 460 voor meer informatie.

3 Koppel de tekst in de gegevensstructuren met de containers en stel opties voor bewerken en bladeren in. Zie "Selecteren en bewerken van tekst en bewerkingen ongedaan maken mogelijk maken met TLF" op pagina 462 voor meer informatie.

4 Maak een gebeurtenislistener om de flow van de tekst aan te passen in reactie op vergroot- of verklein gebeurtenissen (of andere gebeurtenissen). Zie "Gebeurtenissen afhandelen met TLF" op pagina 462 voor meer informatie.

Voorbeeld van Text Layout Framework: nieuwslay-out

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het volgende voorbeeld illustreert het gebruik van het TLF voor de lay-out van een eenvoudige krantenpagina. De pagina bevat een grote kop, een subkop en hoofdtekst die uit meerdere kolommen bestaat:

```
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();
        }
    }
}
```

De klasse TLF gebruikt twee tekst containers. Een container geeft een kop en een subkop weer en de andere container geeft de hoofdtekst van drie kolommen weer. Voor de eenvoud is de tekst in het voorbeeld in code vastgelegd als TFL-opmaak tekst. De variabele `handel` bevat zowel de kop als de subkop en de variabele `opmaakpakket` bevat de hoofdtekst. Zie "Tekst structureren met TLF" op pagina 454 voor meer informatie over TFL Mark up.

Na de initialisatie importeert de functie `onder()` de koptekst in een Text-object, wat de belangrijkste gegevensstructuur van het TLF is:

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

Daarna wordt een Sprite-object gemaakt voor de container en wordt een controller gemaakt en gekoppeld aan de container:

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

De controller is geïnitialiseerd voor het instellen van opties voor opmaken, bladeren en andere bewerkingen. De controller bevat geometrie die de grenzen aangeeft van de container waarin de tekst stroomt. Een Tekstlay-object bevat de opmaakopties:

```
hContainerFormat = new TextLayoutFormat();
```

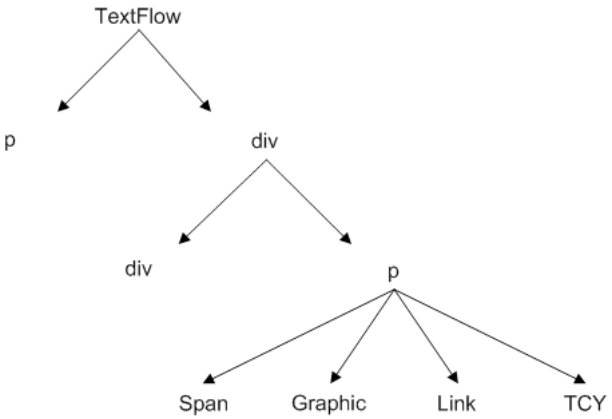
De controller wordt toegewezen aan de onLoadComplete en de functie voegt de container toe aan de weergavelijst. De werkelijke samenstelling en weergave van de containers wordt overgelaten aan de methode `groei()`. Dezelfde stappen worden uitgevoerd om het Text-object van de hoofdtekst te initialiseren.

De methode `resizeHandler()` meet de ruimt die beschikbaar is voor het renderen van de containers en bepaalt de overeenkomstige grootte van de containers. Met het aanvankelijk aanroepen van de methode `compose()` kunt u de correcte hoogte van de kopcontainer berekenen. De methode `resizeHandler()` plaatst vervolgens de kopcontainer en geeft deze weer met de methode `updateAllControllers()`. Ten slotte gebruikt de methode `resizeHandler()` de grootte van de kopcontainer om de plaatsing van de hoofdtekstcontainer te bepalen.

Tekst structureren met TLF

Het TLF maakt gebruik van een hiërarchische structuur om tekst weer te geven. Elk knooppunt in de structuur is een instantie van een klasse die is gedefinieerd in het elementenpakket. Het hoofdknooppunt van de structuur is bijvoorbeeld altijd een instantie van de klasse TextFlow. De klasse TextFlow vertegenwoordigt een heel tekstartikel. Een artikel is een verzameling tekst en andere elementen die als een eenheid of stroom worden behandeld. Voor het weergeven van één artikel zijn mogelijk meerdere kolommen of tekst containers nodig.

Afgezien van het hoofd knooppunt zijn de overige elementen losjes gebaseerd op. Het volgende diagram illustreert de hiërarchie van het framework:



Text-hiërarchie

Text Layout Framework-opmaak

Kennis van de structuur van het TLF is ook nuttig wanneer u met TLF Mark up werkt. TLF Mark up is een van tekst die deel uitmaakt van het TLF. Hoewel het framework ook andere ondersteunt, is de TLF Mark up de enige indeling die specifiek is gebaseerd op de structuur van de Text-hiërarchie. Als u EXAMPLE van een Text exporteert met behulp van deze opmaak indeling, wordt de EXAMPLE met een intacte hiërarchie geëxporteerd.

TLF Mark up biedt de meest getrouwe weergave van tekst in een Text-hiërarchie. De opmaak taal verschaft tag voor alle basiselementen van de Text-hiërarchie naast kenmerken voor alle opmaak eigenschappen in de TextLayoutFormat-klasse.

De volgende tabel bevat de tags die kunnen worden gebruikt in TLF Markup.

Element	**Beschrijving**	**Onderliggende elementen**	**Klasse**
textflow	Het hoofdelement van de Markup.	div, p	TextFlow
div	Een verdeling in een TextFlow. Kan groepen alinea's bevatten.	div, list, p	DivElement
p	Een alinea.	a, tcy, span, img, tab, br, g	ParagraphElement
a	Een koppeling.	tcy, span, img, tab, br, g	LinkElement
tcy	Een stuk horizontale tekst (gebruikt in een verticale TextFlow).	a, span, img, tab, br, g	TCYElement
span	Tekst in een alinea.		SpanElement
img	Een afbeelding in een alinea.		InlineGraphicElement
tab	Een tab-teken.		TabElement
br	Een afbrekingsteken. Wordt gebruikt als regeleinde in een alinea. De tekst loopt verder op de volgende regel, maar blijft in dezelfde alinea.		BreakElement
linkNormalFormat	Bepaalt de opmaakkenmerken die worden gebruikt voor koppelingen in normale toestand.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	Bepaalt de opmaakkenmerken die worden gebruikt voor koppelingen in actieve toestand, wanneer de muis zich op een koppeling bevindt.	TextLayoutFormat	TextLayoutFormat
linkHoverFormat	Bepaalt de opmaakkenmerken die worden gebruikt voor koppelingen in aanwijstoestand, wanneer de muis zich binnen de begrenzingen (schuivend over) van een koppeling bevindt.	TextLayoutFormat	TextLayoutFormat

Element	Beschrijving	Onderliggende elementen	Klasse
li	Een itemelement in een lijst. Moet zich in een lijstelement bevinden.	div, li, list, p	ListItemElement
list	Een lijst. Lijsten kunnen worden genest of naast elkaar worden geplaatst. Er kunnen verschillende labeling- of nummeringsschema's worden toegepast op de items in de lijst.	div, li, list, p	ListElement
g	Een groepselement. Wordt gebruikt voor het groeperen van elementen in een alinea. Hiermee kunt u elementen onder alineaniveau nesten.	a, tcy, span, img, tab, br, g	SubParagraphGroupElement

Meer Help-onderwerpen

TLF 2.0-lijstopmaak

TLF 2.0 SubParagraphGroupElements en typeName

Opsommingslijsten en genummerde lijsten gebruiken

U kunt de klassen ListElement en ListItemElement gebruiken om opsommingstekens toe te voegen aan uw tekstbesturingselementen. Lijsten met opsommingstekens kunnen worden genest en aangepast, zodat verschillende opsommingstekens (of markeringen) en automatische nummering in omtrekstijl worden gebruikt.

Gebruik de tag `<list>` om lijsten op te nemen in uw tekstflow. Vervolgens gebruikt u de tags `<li>` in de tag `<list>` voor elk lijstitem in de lijst. U kunt de weergave van de opsommingstekens aanpassen met gebruik van de klasse ListMarkerFormat.

Met het volgende voorbeeld maakt u eenvoudige lijsten:

```
<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>
```

U kunt lijsten in andere lijsten nesten, zoals uit het volgende voorbeeld blijkt:

```
<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>
```

Als u het type markering in de lijst wilt aanpassen, gebruikt u de eigenschap `listStyleType` van het ListElement. Deze eigenschap kan elke waarde hebben die wordt gedefinieerd door de klasse ListStyleType (zoals `check`, `circle`, `decimal` en `box`). Het volgende voorbeeld maakt lijsten met verschillende typen markeringen en een aangepaste tellerverhoging:

```
<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>
```

Met de klasse ListMarkerFormat definieert u de teller. U kunt niet alleen de verhoging van een teller definiëren, maar u kunt de teller ook aanpassen door deze opnieuw in te stellen met de eigenschap `counterReset`.

U kunt de weergave van de markeringen in uw lijst verder aanpassen met de eigenschappen `beforeContent` en `afterContent` van ListMarkerFormat. Deze eigenschappen zijn van toepassing op inhoud die voor en na de inhoud van de markering wordt weergegeven.

Het volgende voorbeeld voegt de tekenreeks "XX" toe voor de markering en de tekenreeks "YY" na de markering:

```
<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>
```

Met de eigenschap `content` kunt u de opmaak van de markering verder aanpassen. Met het volgende voorbeeld geeft u een geordend Romeins cijfer weer:

```
<flow:list listStyleType="disc"  paddingLeft="96" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="Section "
            content="counters(ordered,&quot;*&quot;,upperRoman)"
            afterContent=": "/>
    </flow:listMarkerFormat>
    <flow:li>Item 1</li>
    <flow:li>Item 2</li>
    <flow:li>Item 3</li>
</flow:list>
```

Zoals uit het vorige voorbeeld blijkt, kunt u met de eigenschap `content` ook een achtervoegsel invoegen: een tekenreeks die na de markering wordt weergegeven, maar vóór `afterContent`. Als u deze tekenreeks wilt invoegen tijdens het verschaffen van de XML-inhoud aan de tekstflow, plaatst u de tekenreeks tussen `&quote;` HTML-entiteiten in plaats van tussen aanhalingstekens "`<string>`").

Meer Help-onderwerpen

TLF 2.0-lijstopmaak

Opvulling gebruiken in TLF

Elk FlowElement biedt ondersteuning voor opvullingseigenschappen waarmee u de positie van het inhoudsgebied van elk element en de ruimte tussen inhoudsgebieden bepaalt.

De totale breedte van een element is het totaal van de breedte van de inhoud plus de eigenschappen `paddingLeft` en `paddingRight`. De totale hoogte van een element is het totaal van de hoogte van de inhoud plus de eigenschappen `paddingTop` en `paddingBottom`.

De opvulling is de ruimte tussen de rand en de inhoud. De opvullingseigenschappen zijn `paddingBottom`, `paddingTop`, `paddingLeft` en `paddingRight`. U kunt vulling toepassen op een TextFlow-object en op de volgende onderliggende elementen:

- div
- img
- li
- list
- p

Opvullingseigenschappen kunnen niet worden toegepast op bereikelementen.

Met het volgende voorbeeld stelt u opvullingseigenschappen in voor de 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">
```

De geldige waarden voor de opvullingseigenschappen zijn een nummer (uitgedrukt in pixels), "auto" of "inherit". De standaardwaarde is "auto". Dat betekent dat de waarde automatisch wordt berekend en voor alle elementen wordt ingesteld op 0, behalve voor ListElement. In geval van ListElements is "auto" 0, behalve aan het begin van de lijst waar de waarde van de eigenschap `listAutoPadding` wordt gebruikt. De standaardwaarde van `listAutoPadding` is 40, wat de standaardinspringing van lijsten is.

Standaard worden opvullingseigenschappen niet overgeërfd. De waarden "auto" en "inherit" zijn constanten die zijn gedefinieerd door de klasse FormatValue.

Opvullingseigenschappen kunnen negatieve waarden zijn.

Meer Help-onderwerpen

Opvullingswijzigingen in TLF 2.0

Tekst opmaken met TLF

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Het pakket flashx.textLayout.formats bevat interfaces en klassen waarmee u indelingen kunt toewijzen aan elk FlowElement in de tekstflowhiërarchiestructuur. Er zijn twee manieren om opmaak toe te passen. U kunt afzonderlijk een specifieke indeling toewijzen of u kunt een groep indelingen tegelijk toewijzen met behulp van een speciaal opmaakobject.

De interface ITextLayoutFormat bevat alle indelingen die op een FlowElement kunnen worden toegepast. Bepaalde indelingen kunnen worden toegepast op een hele container of alinea tekst, maar kunnen niet logisch worden toegepast op afzonderlijke tekens. Indelingen zoals uitvullingen en tabstops bijvoorbeeld kunnen op hele alinea's worden toegepast, maar zijn niet toepasbaar op afzonderlijke tekens.

Indelingen toewijzen aan een FlowElement via eigenschappen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Door eigenschappen toe te wijzen kunt u indelingen instellen op elk FlowElement. De klasse FlowElement implementeert de interface ITextLayoutFormat. Elke subklasse van de FlowElement-klasse moet deze interface dus ook implementeren.

De volgende code toont bijvoorbeeld hoe u afzonderlijke indelingen kunt toewijzen aan een instantie van ParagraphElement:

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

Indelingen toewijzen aan een FlowElement via de TextLayoutFormat-klasse

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de TextLayoutFormat-klasse kunt u indelingen toewijzen aan een FlowElement. Met deze klasse kunt u een speciaal indelingsobject maken waarin alle gewenste indelingswaarden zijn opgenomen. Vervolgens kunt u dit object toewijzen aan de eigenschap `format` van elk FlowElement-object. Zowel met TextLayoutFormat als met FlowElement wordt de ITextLayoutFormat-interface geïmplementeerd. Hiermee wordt verzekerd dat beide klassen dezelfde indelingseigenschappen bevatten.

Zie TextLayoutFormat in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Overerving van indelingen

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Indelingen worden overgeërfd via de tekstflowhiërarchie. Als u een instantie van TextLayoutFormat toewijst aan een FlowElement-instantie met onderliggende niveaus, start het framework een proces dat *trapsgewijs* wordt genoemd. Tijdens een trapsgewijs proces onderzoekt het framework recursief elk knooppunt in de hiërarchie die wordt overgeërfd van uw FlowElement. Vervolgens wordt bepaald of de overgeërfde waarden worden toegewezen aan elke indelingseigenschap. De volgende regels worden toegepast tijdens het trapsgewijze proces:

1. Eigenschapwaarden worden alleen overgeërfd van een directe voorganger (het bovenliggende niveau).
2. Eigenschapwaarden worden alleen overgeërfd wanneer een eigenschap nog geen waarde bevat (de waarde is `undefined`)..
3. Bepaalde kenmerken kunnen geen waarden overerven wanneer deze niet gedefinieerd zijn, tenzij de waarde van het kenmerk is ingesteld op 'inherit' of de constante `flashx.textLayout.formats.FormatValue.INHERIT`.

Als u de waarde `fontSize` bijvoorbeeld op TextFlow-niveau instelt, wordt deze instelling op alle elementen in de TextFlow toegepast. Met andere woorden, de waarden worden trapsgewijs naar beneden doorgegeven door de tekststroomhiërarchie. U kunt echter ook de waarde in een bepaald element overschrijven door een nieuwe waarde rechtstreeks toe te wijzen aan het element. Een tegenvoorbeeld is wanneer u de waarde `backgroundColor` instelt op TextFlow-niveau en de onderliggende elementen van de TextFlow deze waarde niet overerven. De eigenschap `backgroundColor` is een eigenschap die niet wordt overgeërfd van het bovenliggende niveau tijdens een trapsgewijs proces. U kunt dit gedrag opheffen door de eigenschap `backgroundColor` van elk onderliggend element in te stellen op `flashx.textLayout.formats.FormatValue.INHERIT`..

Zie TextLayoutFormat in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Tekst importeren en exporteren met TLF

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de klasse TextConverter in het pakket flashx.textLayout.conversion.* kunt u tekst importeren naar en exporteren uit de TLF. Gebruik deze klasse als u tekst tijdens de runtime wilt laden en de tekst niet in het SWF-bestand wilt compileren. U kunt deze klasse ook gebruiken om tekst die is opgeslagen in een TextFlow-instantie te exporteren naar een String- of XML-object.

Zowel importeren als exporteren is een eenvoudige procedure. U roept eenvoudigweg de methode`export()` of de methode `importToFlow()` aan. Deze beide methoden maken deel uit van de klasse TextConverter. Beide methoden zijn ook statisch, wat betekent dat u de methoden aanroept op de klasse TextConverter in plaats van op een instantie van die klasse.

De klassen in het flashx.textLayout.conversion-pakket geven u veel flexibiliteit bij het kiezen van een opslaglocatie voor uw tekst. Als u de tekst bijvoorbeeld opslaat in een database, kunt u de tekst ter weergave in het framework importeren. U kunt de klassen in het flashx.textLayout.edit-pakket vervolgens gebruiken om de tekst te wijzigen. Daarna exporteert u de gewijzigde tekst terug in uw database.

Zie flashx.textLayout.conversion in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Tekstcontainers beheren met TLF

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als de tekst eenmaal is opgeslagen in TLF-gegevensstructuren, kan deze door Flash Player worden weergegeven. De tekst die is opgeslagen in de flowhiërarchie, moet worden omgezet in een indeling die Flash Player kan weergeven. Het TLF biedt twee manieren om weergaveobjecten te maken uit een flow. De eerste, eenvoudigste manier is geschikt voor het weergeven van statische tekst. De tweede, iets ingewikkelder methode stelt u in staat dynamische tekst te maken die kan worden geselecteerd en bewerkt. In beide gevallen wordt de tekst uiteindelijk omgezet in instanties van de klasse TextLine, die deel uitmaakt van het pakket flash.text.engine.* in Flash Player 10.

Statische tekst maken

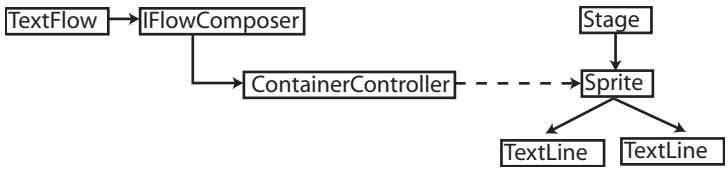
De eenvoudige benadering maakt gebruik van de klasse TextLineFactory, die te vinden is in het flashx.textLayout.factory-pakket. Het voordeel van deze benadering, naast de eenvoud ervan, is dat er minder van het geheugen wordt geëist dan bij de FlowComposer-benadering. Deze benadering kan worden gebruikt voor statische tekst die door de gebruiker niet moet worden bewerkt, geselecteerd of verschoven.

Zie de TextFlowTextLineFactory in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Dynamische tekst en containers maken

Gebruik een flowcomposer als u over meer mogelijkheden wilt beschikken met betrekking tot de weergave van tekst dan TextFlowTextLineFactory biedt. Met een flowcomposer kunnen uw gebruikers de tekst bijvoorbeeld selecteren en bewerken. Zie “Selecteren en bewerken van tekst en bewerkingen ongedaan maken mogelijk maken met TLF” op pagina 462 voor meer informatie.

Een flowcomposer is een instantie van de StandardFlowComposer-klasse in het flashx.textLayout.compose-pakket. Een flowcomposer beheert het omzetten van TextFlow- in TextLine-instanties en het plaatsen van deze TextLine-instanties in een of meer containers.



Een IFlowComposer heeft nul of meer ContainerControllers

Elke TextFlow-instantie heeft een bijbehorend object dat de IFlowComposer-interface implementeert. Dit IFlowComposer-object is toegankelijk via de eigenschap `TextFlow.flowComposer`. U kunt door de IFlowComposer-interface gedefinieerde methoden aanroepen via deze eigenschap. Aan de hand van deze methoden kunt u de tekst koppelen aan een of meerdere containers en de tekst voorbereiden op weergave in een container.

Een container is een instantie van de klasse Sprite, welke een subklasse is van de klasse DisplayObjectContainer. Beide klassen maken deel uit van de weergavelijst-API van Flash Player. Een container is een geavanceerde vorm van het opgegeven kader dat wordt gebruikt met klasse TextLineFactory. Zoals het opgegeven kader definieert een container het gebied waar de TextLine-instantie wordt weergegeven. In tegenstelling tot een selectiekader heeft een container een overeenkomstig "controller"-object. De controller beheert het bladeren, samenstellen, koppelen, opmaken en de gebeurtenisafhandeling van een container of een set containers. Elke container heeft een overeenkomstig controllerobject dat een instantie is van de klasse ContainerController in het pakket flashx.textLayout.container.

Als u tekst wilt weergeven, maakt u een controllerobject dat de container beheert en koppelt u dit aan de flowcomposer. Zodra de container is gekoppeld, stelt u de tekst samen zodat deze kan worden weergegeven. Containers hebben twee statussen: samenstelling en weergave. Tijdens de samenstelling wordt de tekst van een tekststroomhiërarchie geconverteerd in TextLine-instanties en wordt berekend of deze instanties in de container passen. Tijdens weergave wordt de weergavelijst in Flash Player bijgewerkt.

Zie IFlowComposer, StandardFlowComposer en ContainerController in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Selecteren en bewerken van tekst en bewerkingen ongedaan maken mogelijk maken met TLF

Flash Player 9.0 of hoger, Adobe AIR 1.0 of hoger

De mogelijkheid om tekst te selecteren of bewerken wordt beheerd op tekststroomniveau. Elke instantie van de TextFlow-klasse heeft een gekoppeld interactiebeheer. U hebt toegang tot het interactiebeheer van een TextFlow-object via de eigenschap `TextFlow.interactionManager` van het object. Als u tekstselectie wilt inschakelen, wijst u een instantie van de klasse Selection Manager toe aan de eigenschap `interactionManager`. Als u zowel het selecteren als bewerken van tekst wilt mogelijk maken, wijst u een instantie van de klasse EditManager toe in plaats van een instantie van de klasse SelectionManager. Als u bewerkingen voor ongedaan maken wilt mogelijk maken, maakt u een instantie van de klasse UndoManager en voegt u deze toe als argument wanneer u de constructor voor EditManager aanroept. De klasse UndoManager bewaart een geschiedenis van de meest recente bewerkingsactiviteiten van de gebruiker en staat toe dat de gebruiker bepaalde bewerkingen kan ongedaan maken of opnieuw uitvoeren. Deze drie klassen maken deel uit van het editpakket.

Zie SelectionManager, EditManager en UndoManager in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Gebeurtenissen afhandelen met TLF

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

TextFlow-objecten verzenden gebeurtenissen onder vele omstandigheden, waaronder:

- Wanneer de tekst of lay-out wordt gewijzigd
- Voordat een bewerking begint of nadat een bewerking is voltooid
- Wanneer de status van een FlowElement-object verandert
- Wanneer een compositiebewerking is voltooid

Zie flashx.textLayout.events in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Meer Help-onderwerpen

TLF FlowElement, LinkElement Events en EventMirrors

Afbeeldingen in tekst plaatsen

Gebruik de volgende eigenschappen om InlineGraphicElement in de tekst te plaatsen:

- de eigenschap `float` van de klasse InlineGraphicElement
- de eigenschap `clearFloats` van het FlowElement

De eigenschap `float` bepaalt de positie van de afbeelding en de omringende tekst. De eigenschap `clearFloats` bepaalt de positie van de alineaelementen ten opzichte van de `float`.

Gebruik de eigenschap `float` om de locatie van een afbeelding in een tekstelement te bepalen. Het volgende voorbeeld voegt een afbeelding aan een alinea toe en lijnt deze op links uit, zodat de tekst rechts omloopt:

```
<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>
```

De geldige waarden voor de eigenschap `float` zijn "left", "right", "start", "end" en "none". De klasse Float definieert deze constante waarden. De standaardwaarde is "none".

De eigenschap `clearFloats` is nuttig als u het beginpunt van volgende alinea's wilt aanpassen als deze normaliter om de afbeelding zouden lopen. Stel bijvoorbeeld dat u een afbeelding hebt die groter is dan de eerste alinea. Om er zeker van te zijn dat de tweede alinea *na* de afbeelding begint, stelt u de eigenschap `clearFloats` in.

Het volgende voorbeeld gebruikt een afbeelding die groter is dan de tekst in de eerste alinea. Om ervoor te zorgen dat de tweede alinea begint na de afbeelding in het tekstblok, is in dit voorbeeld de eigenschap `clearFloats` voor de tweede alinea ingesteld op "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>
```

De geldige waarden voor de eigenschap `clearFloats` zijn "left", "right", "end", "start", "none" en "both". De klasse ClearFloats definieert deze constanten. U kunt de eigenschap `clearFloats` ook instellen op "inherit", dit is een constante die wordt gedefinieerd door de klasse FormatValue. De standaardwaarde is "none".

Meer Help-onderwerpen

TLF-floats

Hoofdstuk 24: Werken met geluid

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript is bedoeld voor meeslepende, interactieve toepassingen. Bij het maken van dergelijke toepassingen wordt het gebruik van geluid vaak over het hoofd gezien. U kunt geluidseffecten aan een videogame toevoegen, audiofeedback aan de gebruikersinterface van een toepassing toevoegen of zelfs een programma maken voor de analyse van MP3-bestanden die via internet worden geladen, waarbij geluid het hart van de toepassing vormt.

U kunt externe audiobestanden laden en werken met audio die in een SWF-bestand is ingesloten. U kunt de audio te besturen, visuele weergaven van de geluidsinformatie maken en geluid van de microfoon van een gebruiker vastleggen.

Meer Help-onderwerpen

flash.media-pakket

flash.events.SampleDataEvent

Basisbeginselen van het werken met geluid

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Computers kunnen digitale audio vastleggen en coderen. Zo kunnen geluidsgegevens door de computer worden weergegeven, opgeslagen en opgehaald om te worden afgespeeld via luidsprekers. U kunt geluid afspelen met Adobe® Flash® Player of Adobe® AIR™ en ActionScript.

Wanneer geluidsgegevens in een digitale vorm worden omgezet, heeft deze vorm bepaalde kenmerken, zoals het geluidsvolume en het gebruik van stereo- of monogeluid. Wanneer u een geluid in ActionScript afspeelt, kunt u deze kenmerken ook aanpassen: u kunt het geluid harder maken of bijvoorbeeld de indruk wekken dat het uit een bepaalde richting komt.

Voordat u een geluid in ActionScript kunt besturen, moet u de geluidsgegevens eerst in Flash Player of AIR laden. U kunt audiogegevens op vijf manieren in Flash Player of AIR laden om er met behulp van ActionScript mee te kunnen werken.

- U kunt een extern geluidsbestand, zoals een MP3-bestand, in het SWF-bestand laden.
- U kunt de geluidsgegevens tijdens het maken rechtstreeks insluiten in het SWF-bestand.
- U kunt audio vastleggen via een microfoon die is aangesloten op de computer van een gebruiker.
- U kunt audio streamen van een server.
- U kunt audio op dynamische wijze genereren en afspelen.

Wanneer u geluidsgegevens uit een extern geluidsbestand laadt, kunt u starten met het afspelen van het begin van het geluidsbestand terwijl de overige geluidsgegevens nog worden geladen.

Hoewel er uiteenlopende geluidsbestandsindelingen voor codering van digitale audio worden gebruikt, bieden ActionScript 3.0, Flash Player en AIR ondersteuning voor geluidsbestanden die zijn opgeslagen in de MP3-indeling. Geluidsbestanden in andere indelingen, zoals .wav of .aiff, kunnen niet rechtstreeks worden geladen of afgespeeld.

Tijdens het werken met geluid in ActionScript zult u waarschijnlijk met meerdere klassen uit het pakket flash.media werken. De Sound-klasse is de klasse die u gebruikt om toegang tot audiogegevens te krijgen door een geluidsbestand te laden of een functie toe te wijzen aan een gebeurtenis die geluidsgegevens samplet, en vervolgens te beginnen met afspelen. Zodra u met het afspelen van een geluid bent begonnen, bieden Flash Player en AIR u toegang tot een SoundChannel-object. Aangezien een audiobestand dat u hebt geladen, mogelijk slechts een van de vele geluiden is die op de computer van een gebruiker worden afgespeeld, wordt voor elk afzonderlijk geluid dat wordt afgespeeld, een eigen SoundChannel-object gebruikt; de gecombineerde uitvoer van alle SoundChannel-objecten is uiteindelijk wat er via de luidsprekers van de computer wordt afgespeeld. U gebruikt deze SoundChannel-instantie om de eigenschappen van het geluid te besturen en het afspelen te stoppen. Als u de gecombineerde audio wilt besturen, kunt u via de klasse SoundMixer beheer over de gecombineerde uitvoer krijgen.

U kunt nog een aantal andere klassen gebruiken om specifieke taken uit te voeren wanneer u met geluid in ActionScript werkt. Zie "Toelichting bij de geluidsarchitectuur" op pagina 465 voor meer informatie over alle klassen die betrekking hebben op geluid.

Belangrijke concepten en termen

De volgende lijst bevat belangrijke termen die u kunt tegenkomen:

Amplitude De afstand van een punt in de geluidsgolf vanaf de nul- of evenwichtslijn.

Bitsnelheid De hoeveelheid gegevens die voor elke seconde van een geluidsbestand wordt gecodeerd of gestreamd. Bij MP3-bestanden wordt de bitsnelheid meestal aangegeven in duizenden bits per seconden (kbps). Een hogere bitsnelheid correspondeert doorgaans met een geluidsgolf van hogere kwaliteit.

Bufferen Het ontvangen en opslaan van geluidsgegevens voordat deze worden afgespeeld.

mp3 MPEG-1 Audio Layer 3, ofwel MP3, is een populaire indeling voor gecomprimeerd geluid.

Panning De positionering van een audiosignaal tussen het linker- en rechterkanaal in een stereogeluidsveld.

Piek Het hoogste punt in een golf.

Samplingfrequentie Het aantal samples dat per seconden van een analoog audiosignaal wordt genomen om een digitaal signaal te maken. De samplingfrequentie van standaard cd-audio is 44,1 kHz ofwel 44.100 samples per seconde.

Streaming Het proces van het afspelen van de eerdere delen van een geluids- of videobestand terwijl latere delen van dat bestand nog van een server worden geladen.

Volume De sterkte van een geluid.

Golf De vorm van een grafiek van de veranderende amplitudes van een geluidssignaal, uitgezet tegen de tijd.

Toelichting bij de geluidsarchitectuur

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassingen kunnen geluidsgegevens uit vijf hoofdbronnen laden:

- Externe geluidsbestanden die tijdens runtime worden geladen
- Geluidsbronnen die zijn ingesloten in het SWF-bestand van de toepassing
- Geluidsgegevens via een microfoon die op het systeem van de gebruiker is aangesloten
- Geluidsgegevens die worden gestreamd vanaf een externe mediaserver, zoals Flash Media Server

- Geluidsgegevens die dynamisch worden gegenereerd met behulp van de gebeurtenishandler `sampleData`.

Geluidsgegevens kunnen volledig worden geladen voordat deze worden afgespeeld, maar geluidsgegevens kunnen ook worden gestreamd. Dit betekent dat de gegevens worden afgespeeld, terwijl deze nog worden geladen.

De geluidsklassen van ActionScript 3.0 ondersteunen geluidsbestanden die zijn opgeslagen in de MP3-indeling. Geluidsbestanden in andere indelingen, zoals WAV of AIFF, kunnen niet rechtstreeks worden geladen of afgespeeld. Maar vanaf Flash Player 9.0.115.0 kunnen AAC-geluidsbestanden worden geladen en afgespeeld met de klasse NetStream. Dit is dezelfde techniek die wordt gebruikt voor het laden en afspelen van video-inhoud. Zie "Werken met video" op pagina 499 voor meer informatie over deze techniek.

Als u Adobe Flash Professional gebruikt, kunt u WAV- of AIFF-geluidsbestanden importeren en vervolgens in de MP3-indeling in de SWF-bestanden van uw toepassing insluiten. Het Flash-ontwerpgereedschap biedt u ook de mogelijkheid om ingesloten geluidsbestanden te comprimeren om zo hun bestandsgrootte te beperken. Deze inkrimping gaat overigens wel ten koste van de geluidskwaliteit. Zie 'Geluiden importeren' in *Flash gebruiken* voor meer informatie.

De ActionScript 3.0-geluidsarchitectuur maakt gebruik van de volgende klassen uit het pakket flash.media.

Klasse	**Beschrijving**
flash.media.Sound	De Sound-klasse handelt het laden van geluid af, beheert elementaire geluidseigenschappen en start het afspelen van geluid.
flash.media.SoundChannel	Wanneer een toepassing een object Sound afspeelt, wordt een nieuw object SoundChannel gemaakt voor de besturing van het afspelen. Het object SoundChannel bestuurt het volume van het linker- en rechterafspeelkanaal van het geluid. Elk geluid dat wordt afgespeeld, heeft een eigen object SoundChannel.
flash.media.SoundLoaderContext	De klasse SoundLoaderContext bepaalt hoeveel seconden buffering er wordt gebruikt bij het laden van een geluid, en of Flash Player of AIR bij het laden van een bestand zoekt naar een beleidsbestand van de server. Er wordt object een SoundLoaderContext als parameter voor de methode `Sound.load()` gebruikt.
flash.media.SoundMixer	De SoundMixer-klasse bestuurt de afspeel- en de beveiligingseigenschappen van alle geluiden in een toepassing. Meerdere geluidskanalen worden middels een gemeenschappelijk object SoundMixer met elkaar gecombineerd, wat betekent dat eigenschapwaarden in het object SoundMixer betrekking hebben op alle objecten SoundChannel die momenteel worden afgespeeld.
flash.media.SoundTransform	De klasse SoundTransform bevat waarden waarmee geluidsvolume en panning worden bestuurd. Een object SoundTransform kan onder meer op een afzonderlijk object SoundChannel, op het algemene object SoundMixer of op een object Microphone worden toegepast.
flash.media.ID3Info	Een ID3Info-object bevat eigenschappen die ID3-metagegevens vertegenwoordigen. Deze gegevens zijn vaak opgeslagen in MP3-geluidsbestanden.
flash.media.Microphone	De klasse Microphone vertegenwoordigt een microfoon of ander geluidsinvoerapparaat dat op de computer van de gebruiker is aangesloten. Audio-invoer via een microfoon kan naar lokale luidsprekers worden geleid of naar een externe server worden verzonden. Het object Microphone bestuurt de versterking, de bemonsteringsfrequentie en andere kenmerken van de eigen geluidsstream.
flash.media.AudioPlaybackMode	De klasse AudioPlaybackMode definieert constanten voor de eigenschap `audioPlaybackMode` van de klasse SoundMixer.

Elk geluid dat wordt geladen en afgespeeld, heeft zijn eigen instantie van de klasse Sound en de klasse SoundChannel nodig. De uitvoer van meerdere instanties SoundChannel wordt vervolgens tijdens het afspelen met elkaar gecombineerd via de algemene klasse SoundMixer.

De klassen Sound, SoundChannel en SoundMixer worden niet gebruikt voor geluidsgegevens die worden verkregen via een microfoon of vanaf een streaming-mediaserver zoals Flash Media Server.

Externe geluidsbestanden laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elke instantie van de klasse Sound bestaat om een specifieke geluidsbron te laden en het afspelen ervan te starten. Een toepassing kan een object Sound niet hergebruiken om meer dan één geluid te laden. Als er een nieuwe geluidsbron moet worden geladen, moet er een nieuw object Sound worden gemaakt.

Als u een klein geluidsbestand laadt, bijvoorbeeld een klikgeluid dat aan een knop moet worden gekoppeld, kan uw toepassing een nieuw object Sound maken en het geluidsbestand automatisch laten laden, zoals hieronder wordt weergegeven:

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

De constructor `Sound()` accepteert een object URLRequest als eerste parameter. Wanneer er een waarde voor de parameter URLRequest wordt opgegeven, begint het nieuwe object Sound automatisch met het afspelen van de opgegeven geluidsbron.

Uw toepassing moet altijd, behalve in de allersimpelste gevallen, de voortgang van het laden van het geluid in de gaten houden en op fouten tijdens het laden letten. Als het klikgeluid bijvoorbeeld redelijk groot is, is dit wellicht nog niet helemaal geladen wanneer de gebruiker klikt op de knop waarmee het geluid wordt geactiveerd. Als er wordt geprobeerd een geluid af te spelen dat niet is geladen, wordt er mogelijk een runtimefout gegenereerd. Het is beter om te wachten tot het geluid volledig is geladen voordat gebruikers handelingen kunnen uitvoeren waardoor er zou kunnen worden gestart met het afspelen van geluiden.

Een object Sound verzendt een aantal gebeurtenissen tijdens het proces van het laden van een geluid. Uw toepassing kan naar deze gebeurtenissen luisteren om de voortgang van het laden te volgen en ervoor te zorgen dat het geluid volledig is geladen voordat dit wordt afgespeeld. In de volgende tabel worden de gebeurtenissen weergegeven die door een object Sound kunnen worden verzonden.

Gebeurtenis	**Beschrijving**
open (`Event.OPEN`)	Verzonden vlak voordat het laden van geluid wordt gestart.
progress (`ProgressEvent.PROGRESS`)	Periodiek verzonden tijdens het laden van geluid wanneer gegevens van een bestand of stream worden ontvangen.
id3 (`Event.ID3`)	Verzonden wanneer ID3-gegevens beschikbaar zijn voor een MP3-geluid.
complete (`Event.COMPLETE`)	Verzonden wanneer alle gegevens van de geluidsbron zijn geladen.
ioError (`IOErrorEvent.IO_ERROR`)	Verzonden wanneer een geluidsbestand niet is gevonden of wanneer het laadproces wordt onderbroken voordat alle geluidsgegevens zijn ontvangen.

De volgende code laat zien hoe een geluid kan worden afgespeeld nadat het volledig is geladen:

```
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();
}
```

Om te beginnen wordt er in het codevoorbeeld een nieuw object Sound gemaakt, zonder dat er een beginwaarde voor de parameter URLRequest wordt toegewezen. Vervolgens wordt er geluisterd naar de gebeurtenis `Event.COMPLETE` uit het object Sound, die ervoor zorgt dat de methode `onSoundLoaded()` wordt uitgevoerd zodra alle geluidsgegevens zijn geladen. Daarna wordt de methode `Sound.load()` aangeroepen met een nieuwe URLRequest-waarde voor het geluidsbestand.

De methode `onSoundLoaded()` wordt uitgevoerd zodra het geluid volledig is geladen. De eigenschap `target` van het object Event is een verwijzing naar het object Sound. Door het aanroepen van de methode `play()` van het object Sound wordt vervolgens het afspelen van het geluid gestart.

Het proces van het laden van geluiden volgen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Geluidsbestanden kunnen zeer groot zijn en het laden ervan kan veel tijd kosten. Hoewel Flash Player en AIR toestaan dat uw toepassing geluiden afspeelt voordat deze volledig zijn geladen, kan het nuttig zijn om de gebruiker te laten weten hoeveel er al van de geluidsgegevens is geladen en hoeveel van het geluid al is afgespeeld.

De klasse Sound verzendt twee gebeurtenissen die ervoor zorgen dat u de voortgang van het laden van een geluid redelijk gemakkelijk kunt weergeven: `ProgressEvent.PROGRESS` en `Event.COMPLETE`. In het volgende voorbeeld ziet u hoe u deze gebeurtenissen kunt gebruiken om voortgangsgegevens weer te geven over het geluid dat wordt geladen:

```
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);
}
```

In deze code wordt eerst een object Sound gemaakt, waarna er listeners voor de gebeurtenissen `ProgressEvent.PROGRESS` en `Event.COMPLETE` aan dat object worden toegevoegd. Nadat de methode `Sound.load()` is aangeroepen en de eerste gegevens van het geluidsbestand zijn ontvangen, treedt de gebeurtenis `ProgressEvent.PROGRESS` op en wordt de methode `onSoundLoadProgress()` geactiveerd.

Het percentage geladen geluidsgegevens is gelijk aan de waarde van de eigenschap `bytesLoaded` van het object ProgressEvent gedeeld door de waarde van de eigenschap `bytesTotal`. De eigenschappen `bytesLoaded` en `bytesTotal` zijn ook beschikbaar voor het object Sound. In het bovenstaande voorbeeld worden alleen berichten over de voortgang van het laden van geluid weergegeven, maar u kunt de waarden `bytesLoaded` en `bytesTotal` ook heel gemakkelijk gebruiken om voortgangsbalken bij te werken, die bijvoorbeeld bij het framework van Adobe Flex of het Adobe Flash-programma voor het schrijven van programmacode worden geleverd.

In dit voorbeeld ziet u ook hoe een toepassing bij het laden van geluidsbestanden een fout kan detecteren en erop kan reageren. Als een geluidsbestand met een opgegeven bestandsnaam niet wordt gevonden, wordt bijvoorbeeld de gebeurtenis `Event.IO_ERROR` door het object Sound verzonden. In bovenstaande code wordt de methode `onIOError()` uitgevoerd en wordt een kort foutbericht weergegeven wanneer een fout optreedt.

Werken met ingesloten geluiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het gebruik van ingesloten geluiden, in plaats van het laden van geluiden uit een extern bestand, is vooral nuttig bij kleine geluiden die worden gebruikt als indicatoren binnen de gebruikersinterface van uw toepassing, bijvoorbeeld geluiden die worden afgespeeld wanneer er op knoppen wordt geklikt.

Wanneer u een geluidsbestand in uw toepassing insluit, wordt het resulterende SWF-bestand vergroot met het geluidsbestand. Met andere woorden, het insluiten van grote geluidsbestanden in uw toepassing kan ertoe leiden dat uw SWF-bestand ongewenst groot wordt.

Afhankelijk van uw ontwikkelingsomgeving, verschilt de exacte methode waarop een geluidsbestand in het SWF-bestand van uw toepassing wordt ingesloten.

Een ingesloten geluidsbestand in Flash gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met het Flash-ontwerpgereedschap kunt u geluiden in diverse geluidsindelingen importeren en deze opslaan als symbolen in de bibliotheek. U kunt de geluiden vervolgens aan frames in de tijdlijn of aan de frames van een knoptoestand toewijzen, de geluiden met gedragingen gebruiken of deze rechtstreeks in ActionScript-code gebruiken. In deze sectie wordt beschreven hoe u met behulp van het Flash-ontwerpgereedschap ingesloten geluiden in ActionScript-code kunt gebruiken. Zie 'Geluiden importeren' in *Flash gebruiken* voor informatie over andere manieren om ingesloten geluiden in Flash te gebruiken.

U kunt als volgt een geluidsbestand insluiten met behulp van het Flash-programma voor het schrijven van programmacode:

1. Selecteer Bestand > Importeren > Importeren in bibliotheek, selecteer een geluidsbestand en importeer dit.
2. Klik met de rechtermuisknop op de naam van het geïmporteerde bestand in het deelvenster Bibliotheek en selecteer Eigenschappen. Klik op het selectievakje Exporteren voor ActionScript.
3. Voer in het veld Klasse een naam in voor gebruik bij verwijzing naar dit ingesloten geluid in ActionScript. Standaard wordt de naam van het geluidsbestand in dit veld gebruikt. Als de bestandsnaam een punt bevat, zoals in de naam "DrumSound.mp3", moet u de naam veranderen, bijvoorbeeld in "DrumSound"; ActionScript staat namelijk geen punt in de naam van een klasse toe. In het veld Basisklasse wordt nog steeds flash.media.Sound weergegeven.
4. Klik op OK. Er wordt mogelijk een dialoogvenster weergegeven waarin wordt gemeld dat er geen definitie voor deze klasse in het klassenpad is gevonden. Klik op OK en ga verder. Als u een klassennaam hebt ingevoerd die niet overeenkomt met de naam van een van de klassen in het klassenpad van uw toepassing, wordt er automatisch een nieuwe klasse gegenereerd die de eigenschappen overerft van de klasse flash.media.Sound.
5. Als u het ingesloten geluid wilt gebruiken, verwijst u naar de klassennaam voor dat geluid in ActionScript. De volgende code begint bijvoorbeeld met het maken van een nieuwe instantie van de automatisch gegenereerde klasse DrumSound:

   ```
   var drum:DrumSound = new DrumSound();
   var channel:SoundChannel = drum.play();
   ```

   DrumSound is een subklasse van de klasse flash.media.Sound, wat betekent dat de methoden en eigenschappen van de klasse Sound worden overgeërfd, inclusief de methode `play()`, zoals hierboven is weergegeven.

Een ingesloten geluidsbestand in Flex gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn veel manieren waarop u geluidselementen in een Flex-toepassing kunt insluiten, zoals:

- Met de metagegevenstag `[Embed]` in een script

- Door met de aanwijzing `@Embed` in MXML een ingesloten geluidselement toe te wijzen als een eigenschap van een onderdeel zoals een Button of een SoundEffect.
- Met de aanwijzing `@Embed` binnen een CSS-bestand

In deze sectie wordt de eerste optie beschreven: geluiden insluiten in ActionScript-code binnen een Flex-toepassing met behulp van de metagegevenstag `[Embed]`.

Als u een element wilt insluiten in ActionScript-code, gebruikt u de metagegevenstag `[Embed]`.

Plaats het geluidsbestand in de hoofdbronmap of in een andere map die zich bevindt in het bouwpad van het project. Wanneer de compiler van een metagegevenstag Embed aantreft, wordt het ingesloten klassenelement automatisch gemaakt. U krijgt toegang tot de klasse via een variabele van het gegevenstype Class die u direct na de metagegevenstag `[Embed]` moet declareren.

Met de volgende code wordt een geluid met de naam smallSound.mp3 ingesloten en wordt een variabele met de naam `soundClass` gebruikt om een verwijzing op te slaan naar het ingesloten klassenelement dat aan het geluid is gekoppeld. De code maakt vervolgens een instantie van het ingesloten klassenelement, cast deze als een instantie van de klasse Sound en roept de methode `play()` aan voor die instantie:

```
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();
        }
    }
}
```

Als u het ingesloten geluid wilt gebruiken om een eigenschap van een Flex-component in te stellen, moet u het geluid casten als een instantie van de klasse mx.core.SoundAsset en niet als een instantie van de klasse Sound. Zie "Ingesloten klassenelementen" in ActionScript 3.0 leren gebruiken voor een vergelijkbaar voorbeeld waarin de SoundAsset-klasse wordt gebruikt.

Werken met gestreamde geluidsbestanden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer een geluidsbestand of videobestand wordt afgespeeld terwijl er nog gegevens uit het bestand worden geladen, is er sprake van *streaming*. Externe geluidsbestanden die vanaf een externe server worden geladen, worden vaak gestreamd, zodat de gebruiker niet hoeft te wachten tot alle geluidsgegevens zijn geladen om naar het geluid te kunnen luisteren.

De eigenschap `SoundMixer.bufferTime` staat voor het aantal milliseconden aan geluidsgegevens dat door Flash Player of AIR moet worden verzameld voordat er kan worden gestart met het afspelen van het geluid. Met andere woorden, als de eigenschap `bufferTime` op 5000 is ingesteld, laadt Flash Player of AIR ten minste 5000 milliseconden aan gegevens uit het geluidsbestand voordat het afspelen van het geluid wordt gestart. De standaardwaarde voor `SoundMixer.bufferTime` is 1000.

Uw toepassing kan de algemene waarde van `SoundMixer.bufferTime` voor een specifiek geluidsbestand onderdrukken door bij het laden van het geluid expliciet een nieuwe waarde voor `bufferTime` op te geven. Als u de standaardbuffertijd wilt onderdrukken, maakt u eerst een nieuwe instantie van de klasse SoundLoaderContext, stelt u de waarde voor de eigenschap `bufferTime` hiervan in en geeft u dit vervolgens door als parameter aan de methode `Sound.load()`, zoals hieronder wordt weergegeven:

```
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();
```

Terwijl het afspelen wordt vervolgd, probeert Flash Player of AIR de geluidsbuffer even groot of groter te houden. Als de geluidsgegevens sneller dan de afspeelsnelheid worden geladen, wordt het afspelen zonder onderbreking voortgezet. Als de snelheid waarmee de gegevens worden geladen door netwerkbeperkingen afneemt, zou de afspeelkop echter het einde van de geluidsbuffer kunnen bereiken. Als dit gebeurt, wordt het afspelen onderbroken, om vervolgens automatisch te worden hervat zodra er meer geluidsgegevens zijn geladen.

Als u wilt nagaan of het afspelen is onderbroken omdat Flash Player of AIR wacht tot er gegevens zijn geladen, gebruikt u de eigenschap `Sound.isBuffering`.

Werken met dynamisch gegenereerde audio

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Opmerking: *De mogelijkheid om geluid dynamisch te genereren, is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5.*

In plaats van een bestaand geluid te laden of te streamen, kunt u audiogegevens dynamisch genereren. U kunt audiogegevens genereren wanneer u een gebeurtenislistener toewijst voor de gebeurtenis `sampleData` van een Sound-object. (De gebeurtenis `sampleData` wordt gedefinieerd in de klasse SampleDataEvent in het pakket flash.events.) In deze omgeving laadt het Sound-object geen geluidsgegevens uit een bestand. Het fungeert als socket voor geluidsgegevens die ernaar worden gestreamd door het gebruik van de functie die u aan deze gebeurtenis toewijst.

Wanneer u de gebeurtenislistener `sampleData` toevoegt aan een Sound-object, vraagt het object regelmatig om gegevens die worden toegevoegd aan de geluidsbuffer. Deze buffer bevat gegevens die door het Sound-object worden afgespeeld. Wanneer u de methode `play()` van het Sound-object oproept, verzendt deze de gebeurtenis `sampleData` als nieuwe geluidsgegevens worden aangevraagd. (Dit is alleen van toepassing als het Sound-object geen MP3-gegevens heeft geladen uit een bestand.)

Het SampleDataEvent-object bevat de eigenschap `data`. In de gebeurtenislistener schrijft u ByteArray-objecten naar dit `data`-object. De bytearrays die u naar dit object schrijft, worden toegevoegd aan gebufferde geluidsgegevens die door het Sound-object worden afgespeeld. De bytearray in de buffer is een stream drijvende-kommawaarden van -1 tot 1. Elke drijvende-kommawaarde vertegenwoordigt de amplitude van één kanaal (links of rechts) van een geluidssample. Geluid wordt gesampled met 44.100 samples per seconde. Elke sample bevat een linker- en rechterkanaal en is interleaved als drijvende-kommagegevens in de bytearray.

In uw handlerfunctie gebruikt u de methode `ByteArray.writeFloat()` om de eigenschap `data` naar de gebeurtenis `sampleData` te schrijven. De volgende code genereert bijvoorbeeld een sinusgolf:

```
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);
    }
}
```

Wanneer u `Sound.play()` oproept, begint de toepassing de gebeurtenishandler op te roepen en gegevens met geluidssamples te vragen. De toepassing blijft gebeurtenissen verzenden terwijl het geluid wordt afgespeeld, totdat u stopt met het aanleveren van gegevens of totdat u `SoundChannel.stop()` oproept.

De vertraging van gebeurtenissen varieert per platform en kan veranderen in toekomstige versies van Flash Player en AIR. Het is verstandig niet uit te gaan van een specifieke vertraging, maar deze te berekenen. Gebruik de volgende formule om de vertraging te berekenen:

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

Lever 2048 tot 8192 samples aan de eigenschap `data` van het SampleDataEvent-object (voor elke oproep van de gebeurtenislistener). U krijgt de beste resultaten als u zoveel mogelijk samples aanlevert (maximaal 8192). Hoe minder samples u aanlevert, des te groter de kans op ongewenste geluiden en onderbrekingen tijdens het afspelen. Dit gedrag verschilt per platform en kan in verschillende situaties optreden, bijvoorbeeld wanneer u de afmetingen van de browser wijzigt. Code die goed werkt op het ene platform wanneer u slechts 2048 samples aanlevert, kan mindere resultaten geven op een ander platform. Als u de kleinst mogelijke vertraging nodig hebt, kunt u overwegen om de hoeveelheid gegevens door de gebruiker te laten selecteren.

Als u minder dan 2048 samples aanlevert (per oproep van de gebeurtenislistener `sampleData`), stopt de toepassing na het afspelen van de resterende samples. Het SoundChannel-object verzendt dan een SoundComplete-gebeurtenis.

Geluid wijzigen van MP3-gegevens

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Met de methode `Sound.extract()` haalt u gegevens op van een Sound-object. U kunt deze gegevens gebruiken (en wijzigen) om te schrijven naar de dynamische stream van een Sound-object voor afspelen. In de volgende code worden bijvoorbeeld de bytes van een geladen MP3-bestand gebruikt en doorgegeven via de filterfunctie `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;
}
```

Beperkingen betreffende gegenereerde geluiden

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

Als u de gebeurtenislistener `sampleData` gebruikt in combinatie met een Sound-object, worden alleen de volgende methoden van Sound ingeschakeld: `Sound.extract()` en `Sound.play()`. Als u andere methoden of eigenschappen oproept, resulteert dit in een uitzonderingsfout. Alle methoden en eigenschappen van het object SoundChannel blijven ingeschakeld.

Geluiden afspelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het afspelen van een geladen geluid kan eenvoudig worden bereikt door de methode `Sound.play()` voor een object Sound aan te roepen. Dit doet u als volgt:

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

Bij het afspelen van geluiden met behulp van ActionScript 3.0 kunt u de volgende bewerkingen uitvoeren:

- Een geluid afspelen vanaf een specifieke startpositie

- Een geluid onderbreken en het afspelen later vanaf hetzelfde punt hervatten
- Precies nagaan wanneer het afspelen van een geluid wordt voltooid
- De voortgang van het afspelen van een geluid volgen
- Volume of panning wijzigen tijdens het afspelen van een geluid

Als u deze bewerkingen tijdens het afspelen wilt uitvoeren, gebruikt u de klassen SoundChannel, SoundMixer en SoundTransform.

De klasse SoundChannel bestuurt het afspelen van één geluid. De eigenschap `SoundChannel.position` kan worden beschouwd als een afspeelkop, waarmee de huidige positie binnen de afgespeelde geluidsgegevens wordt aangegeven.

Wanneer een toepassing de methode `Sound.play()` aanroept, wordt een nieuwe instantie van de klasse SoundChannel gemaakt om het afspelen te besturen.

Uw toepassing kan een geluid vanaf een specifieke startpositie afspelen door die positie, in milliseconden, door te geven als waarde voor de parameter `startTime` van de methode `Sound.play()`. U kunt ook opgeven dat het geluid een vast aantal maal snel na elkaar moet worden herhaald, door een numerieke waarde in de parameter `loops` van de methode `Sound.play()` door te geven.

Wanneer de methode `Sound.play()` wordt aangeroepen met zowel de parameter `startTime` als de parameter `loops`, wordt het geluid herhaaldelijk vanaf hetzelfde startpunt afgespeeld, zoals wordt weergegeven in de volgende code:

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

In dit voorbeeld wordt het geluid drie keer achter elkaar afgespeeld vanaf een punt dat één seconde na het begin van het geluid ligt.

Een geluid onderbreken en hervatten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als uw toepassing lange geluiden zoals liedjes of podcasts afspeelt, wilt u gebruikers waarschijnlijk de mogelijkheid bieden om het afspelen van deze geluiden te onderbreken en te hervatten. Een geluid kan tijdens het afspelen in ActionScript niet echt worden onderbroken, maar alleen worden gestopt. Een geluid kan echter vanaf elk gewenst punt worden afgespeeld. U kunt de positie van het geluid vastleggen op het moment van stoppen, en het geluid vervolgens later vanaf die positie weer afspelen.

Stel bijvoorbeeld dat in uw code een geluidsbestand als volgt wordt geladen en afgespeeld:

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

Terwijl het geluid wordt afgespeeld, geeft de eigenschap `SoundChannel.position` steeds het punt in het geluidsbestand aan dat op dat moment wordt afgespeeld. Uw toepassing kan als volgt de positiewaarde opslaan voordat wordt gestopt met het afspelen van het geluid:

```
var pausePosition:int = channel.position;
channel.stop();
```

Als u het afspelen van het geluid wilt hervatten, geeft u de eerder opgeslagen positiewaarde door om het geluid weer te starten vanaf het punt waar het afspelen eerder is gestopt.

```
channel = snd.play(pausePosition);
```

De status van het afspelen volgen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassing wil mogelijk weten wanneer het afspelen van een geluid wordt gestopt, om met het afspelen van een ander geluid te kunnen beginnen, of om resources vrij te maken die tijdens de eerdere afspeelbewerking werden gebruikt. De klasse SoundChannel verzendt de gebeurtenis `Event.SOUND_COMPLETE` wanneer het afspelen van het corresponderende geluid is voltooid. Uw toepassing kan naar deze gebeurtenis luisteren en op basis daarvan actie ondernemen, zoals hieronder wordt weergegeven:

```
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.");
}
```

De klasse SoundChannel verzendt tijdens het afspelen geen voortgangsgebeurtenissen. Om melding te kunnen maken van de voortgang van het afspelen, kunt u een eigen tijdsmechanisme voor uw toepassing opzetten en de positie van de geluidsafspeelkop bijhouden.

Als u wilt berekenen welk percentage van een geluid is afgespeeld, deelt u de waarde van de eigenschap `SoundChannel.position` door de lengte van de geluidsgegevens die worden afgespeeld:

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

Met deze code worden echter alleen nauwkeurige afspeelpercentages gemeld als de geluidsgegevens volledig geladen waren voordat met afspelen werd begonnen. De eigenschap `Sound.length` geeft de grootte van de momenteel geladen geluidsgegevens aan, niet de uiteindelijke grootte van het hele geluidsbestand. Als u de voortgang wilt bijhouden van het afspelen van een streaming geluid dat nog steeds wordt afgespeeld, moet uw toepassing een schatting maken van de uiteindelijke grootte van het hele geluidsbestand en deze waarde in de berekeningen gebruiken. U kunt een schatting van de uiteindelijke lengte van de geluidsgegevens maken door als volgt gebruik te maken van de eigenschappen `bytesLoaded` en `bytesTotal`:

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

Met de volgende code wordt een groter geluidsbestand geladen en wordt de gebeurtenis `Event.ENTER_FRAME` gebruikt als tijdsmechanisme voor weergave van de voortgang van het afspelen. Het afspeelpercentage wordt periodiek gemeld; dit wordt berekend als de huidige positiewaarde gedeeld door de totale lengte van de geluidsgegevens:

```
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);
}
```

Nadat is gestart met het laden van de geluidsgegevens, roept deze code de methode `snd.play()` aan en wordt het resulterende object SoundChannel opgeslagen in de variabele `channel`. Vervolgens wordt er een gebeurtenislistener aan de hoofdtoepassing toegevoegd voor de gebeurtenis `Event.ENTER_FRAME`, en wordt een gebeurtenislistener aan het object SoundChannel toegevoegd voor de gebeurtenis `Event.SOUND_COMPLETE`, die optreedt zodra het afspelen is voltooid.

Steeds wanneer de toepassing een nieuw frame binnen de animatie bereikt, wordt de methode `onEnterFrame()` aangeroepen. De methode `onEnterFrame()` maakt een schatting van de totale lengte van het geluidsbestand op basis van de hoeveelheid gegevens die al zijn geladen, waarna wordt berekend en weergegeven welk percentage al is afgespeeld.

Wanneer het hele geluid is afgespeeld, wordt de methode `onPlaybackComplete()` uitgevoerd en wordt de gebeurtenislistener voor de gebeurtenis `Event.ENTER_FRAME` verwijderd, zodat er niet wordt geprobeerd om de voortgang van het afspelen te blijven bijwerken als het afspelen eenmaal is voltooid.

De gebeurtenis `Event.ENTER_FRAME` kan een groot aantal maal per seconde worden verzonden. In sommige gevallen zult u de voortgang van het afspelen niet zo vaak willen weergeven. In die gevallen kunt u een eigen tijdsmechanisme voor uw toepassing opzetten met behulp van de klasse flash.util.Timer; zie “Werken met datums en tijden” op pagina 1.

Gestreamde geluiden stoppen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het afspeelproces voor gestreamde geluiden kent een eigenaardigheid. Gestreamde geluiden zijn geluiden die nog worden geladen terwijl ze al worden afgespeeld. Wanneer uw toepassing de methode `SoundChannel.stop()` aanroept voor een instantie SoundChannel die een streaming geluid afspeelt, wordt het afspelen van het geluid voor één frame stopgezet, waarna er bij het volgende frame weer vanaf het begin van het geluid wordt gestart. Dit gebeurt omdat het proces voor het laden van het geluid nog actief is. Als u zowel het laden als het afspelen van een streaming geluid wilt stoppen, roept u de methode `Sound.close()` aan.

Beveiligingsoverwegingen bij het laden en afspelen van geluiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De mogelijkheden van uw toepassing tot het verkrijgen van toegang tot geluidsgegevens worden mogelijk beperkt door het beveiligingsmodel van Flash Player of AIR. Elk geluid is onderhevig aan de beperkingen van twee verschillende beveiligingssandboxen: de sandbox voor de inhoud zelf (de 'inhoudssandbox') en de sandbox voor de toepassing of het object waarmee het geluid wordt geladen en afgespeeld (de 'eigenaarssandbox'). In het geval van inhoud van AIR in de sandbox met toepassingsbeveiliging zijn alle geluiden, inclusief de geluiden die uit andere domeinen werden geladen, toegankelijk voor inhoud in de sandbox met toepassingsbeveiliging. Voor inhoud in andere beveiligingssandboxen gelden echter dezelfde regels als voor inhoud die wordt uitgevoerd in Flash Player. Zie “Beveiliging” op pagina 1106 voor meer informatie over het beveiligingsmodel van Flash Player in het algemeen, en de definitie van sandboxen.

De inhoudssandbox bepaalt of gedetailleerde geluidsgegevens uit het geluid kunnen worden geëxtraheerd met behulp van de eigenschap `id3` of de methode `SoundMixer.computeSpectrum()`. Deze sandbox legt geen beperkingen op aan het laden of afspelen van het geluidsbestand zelf.

Het oorspronkelijke domein van het geluidsbestand bepaalt de beveiligingsbeperkingen van de inhoudssandbox. Als een geluidsbestand zich in hetzelfde domein of dezelfde map bevindt als het SWF-bestand van de toepassing of het object waarmee het bestand wordt geladen, heeft de toepassing of het object volledige toegang tot dat geluidsbestand. Als het geluid uit een ander domein komt dan de toepassing, kan dit nog steeds in de inhoudssandbox worden gebracht door middel van een beleidsbestand.

Uw toepassing kan een SoundLoaderContext-object met de eigenschap `checkPolicyFile` als parameter doorgeven aan de methode `Sound.load()`. Als de eigenschap `checkPolicyFile` wordt ingesteld op `true`, weet Flash Player of AIR dat er moet worden gezocht naar een beleidsbestand op de server waarvan het geluid wordt geladen. Als er een beleidsbestand bestaat en dit bestand toegang verleent tot het domein van het SWF-bestand dat wordt geladen, kan het SWF-bestand het geluidsbestand laden, toegang krijgen tot de eigenschap `id3` van het Sound-object en de methode `SoundMixer.computeSpectrum()` voor geladen geluiden oproepen.

De eigenaarssandbox bestuurt het lokale afspelen van de geluiden. De toepassing of het object waarmee het afspelen van een geluid wordt gestart, definieert de eigenaarssandbox.

Met de methode `SoundMixer.stopAll()` worden de geluiden gedempt in alle objecten SoundChannel die momenteel worden afgespeeld, mits deze aan de volgende criteria voldoen:

- De geluiden zijn gestart door objecten met dezelfde eigenaarssandbox.
- De geluiden zijn afkomstig uit een bron met een beleidsbestand dat toegang verleent tot het domein van de toepassing of het object waarmee de methode `SoundMixer.stopAll()` wordt opgeroepen.

In een AIR-toepassing wordt de inhoud in de beveiligingssandbox van de toepassing (inhoud die met de AIR-toepassing is geïnstalleerd) echter niet beperkt door deze beveiligingsbeperkingen.

Om na te gaan of de methode `SoundMixer.stopAll()` inderdaad alle geluiden die worden afgespeeld zal stopzetten, kan uw toepassing de methode `SoundMixer.areSoundsInaccessible()` aanroepen. Als die methode de waarde `true` retourneert, betekent dit dat een deel van de geluiden die worden afgespeeld, buiten de controle van de huidige eigenaarssandbox vallen en niet worden stopgezet door de methode `SoundMixer.stopAll()`.

Met de methode `SoundMixer.stopAll()` wordt de afspeelkop gestopt, zodat geen van de geluiden die uit externe bestanden zijn geladen, verder kunnen worden afgespeeld. Geluiden die zijn ingesloten in FLA-bestanden en met behulp van het Flash-ontwerpgereedschap aan frames in de tijdlijn zijn gekoppeld, worden mogelijk echter weer afgespeeld als de animatie naar een nieuw frame wordt verplaatst.

Volume en panning voor geluid instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Eén specifiek object SoundChannel bestuurt zowel het linker- als rechterstereokanaal voor een geluid. Als een MP3-geluid een monogeluid is, bevatten het linker- en rechterstereokanaal van het object SoundChannel identieke geluidsgolven.

U kunt de amplitude van elk stereokanaal van het geluid dat wordt afgespeeld, opvragen via de eigenschappen `leftPeak` en `rightPeak` van het object SoundChannel. Deze eigenschappen geven de piekamplitude van de geluidsgolf zelf aan. Hiermee wordt niet het feitelijke afspeelvolume aangegeven. Het feitelijke afspeelvolume is een functie van de amplitude van de geluidsgolf en de volumewaarden die zijn ingesteld in het object SoundChannel en de klasse SoundMixer.

De eigenschap pan van een object SoundChannel kan worden gebruikt om aparte volumeniveaus op te geven voor het afspelen van het linker- en rechterkanaal. De eigenschap pan kan een waarde tussen -1 en 1 hebben, waarbij -1 betekent dat het linkerkanaal met maximaal volume wordt afgespeeld terwijl het rechterkanaal wordt gedempt, en 1 betekent dat het rechterkanaal met maximaal volume wordt afgespeeld terwijl het linkerkanaal wordt gedempt. Met numerieke waarden tussen –1 en 1 worden proportionele waarden voor de waarden van het linker- en rechterkanaal ingesteld, en bij de waarde 0 worden beide kanalen met hetzelfde, middelhoge volumeniveau afgespeeld.

In het volgende voorbeeld wordt een object SoundTransform met volumewaarde 0,6 en panwaarde -1 gemaakt (volume van linkerkanaal maximaal en rechterkanaal gedempt). Het object SoundTransform wordt doorgegeven als parameter aan de methode `play()`, waarmee dat object SoundTransform wordt toegepast op het nieuwe object SoundChannel dat wordt gemaakt ter besturing van het afspelen.

```
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);
```

U kunt volume en panning wijzigen terwijl een geluid wordt afgespeeld door de eigenschap `pan` of `volume` van een object `soundTransform` in te stellen en dat object vervolgens toe te passen als de eigenschap soundTransform van een object SoundChannel.

U kunt ook algemene waarden voor volume en pan instellen die voor alle geluiden tegelijk worden gebruikt, door gebruik te maken van de eigenschap `soundTransform` van de klasse SoundMixer. In het volgende voorbeeld ziet u hoe u dit kunt doen:

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

U kunt ook een object SoundTransform gebruiken om waarden voor volume en pan in te stellen voor een object Microphone (zie "Geluidsinvoer vastleggen" op pagina 485) en voor objecten Sprite en SimpleButton.

In het volgende voorbeeld wordt panning van het geluid afgewisseld van het linker- naar het rechterkanaal en terug, terwijl het geluid wordt afgespeeld.

```
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);
}
```

Deze code begint met het laden van een geluidsbestand, waarna er een nieuw object SoundTransform wordt gemaakt met volume ingesteld op 1 (maximaal volume) en pan ingesteld op 0 (gelijkmatige verdeling tussen links en rechts). Daarna wordt de methode `snd.play()` aangeroepen, waarbij het object SoundTransform als parameter wordt doorgegeven.

Terwijl het geluid wordt afgespeeld, wordt herhaaldelijk de methode `onEnterFrame()` uitgevoerd. De methode `onEnterFrame()` gebruikt de functie `Math.sin()` om een waarde tussen -1 en 1 te genereren, zodat de waarde binnen het bereik van aanvaardbare waarden voor de eigenschap `SoundTransform.pan` valt. De eigenschap `pan` van het object SoundTransform wordt ingesteld op de nieuwe waarde, waarna de eigenschap `soundTransform` van het kanaal zodanig wordt ingesteld dat het aangepaste object SoundTransform wordt gebruikt.

Als u dit voorbeeld wilt uitvoeren, vervangt u de bestandsnaam bigSound.mp3 door de naam van een lokaal MP3-bestand. Vervolgens voert u het voorbeeld uit. U zou dan het volume van het linkerkanaal hoger moeten horen worden, terwijl het volume van het rechterkanaal lager wordt, en omgekeerd.

In dit voorbeeld kan hetzelfde effect worden bereikt door de eigenschap `soundTransform` van de klasse SoundMixer in te stellen. Dit zou echter invloed hebben op de panning van alle geluiden die op dat moment worden afgespeeld, en niet alleen op het geluid dat door dit object SoundChannel wordt afgespeeld.

Werken met geluidsmetagegevens

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Geluidsbestanden die de MP3-indeling gebruiken, kunnen aanvullende gegevens over het geluid bevatten, in de vorm van ID3-tags.

Niet alle MP3-bestanden bevatten ID3-metagegevens. Wanneer een object Sound een MP3-geluidsbestand laadt, wordt de gebeurtenis `Event.ID3` verzonden als het geluidsbestand ID3-metagegevens bevat. Om uitvoeringsfouten te voorkomen, moet uw toepassing wachten tot de gebeurtenis `Event.ID3` is ontvangen voordat de eigenschap `Sound.id3` voor een geladen bestand wordt opgevraagd.

In de volgende code ziet u hoe u kunt merken of de ID3-metagegevens voor een geluidsbestand zijn geladen:

```
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]);
    }
}
```

Deze code begint met het maken van een object Sound, waarna wordt geluisterd naar de gebeurtenis `Event.ID3`. Wanneer de ID3-metagegevens van het geluidsbestand zijn geladen, wordt de methode `onID3InfoReceived()` aangeroepen. Het doel van het object Event dat wordt doorgegeven aan de methode `onID3InfoReceived()`, is het oorspronkelijke object Sound, wat betekent dat de methode vervolgens de eigenschap `id3` van het object Sound ontvangt en dan alle benoemde eigenschappen doorloopt om de bijbehorende waarden op te vragen.

Onbewerkte geluidsgegevens opvragen

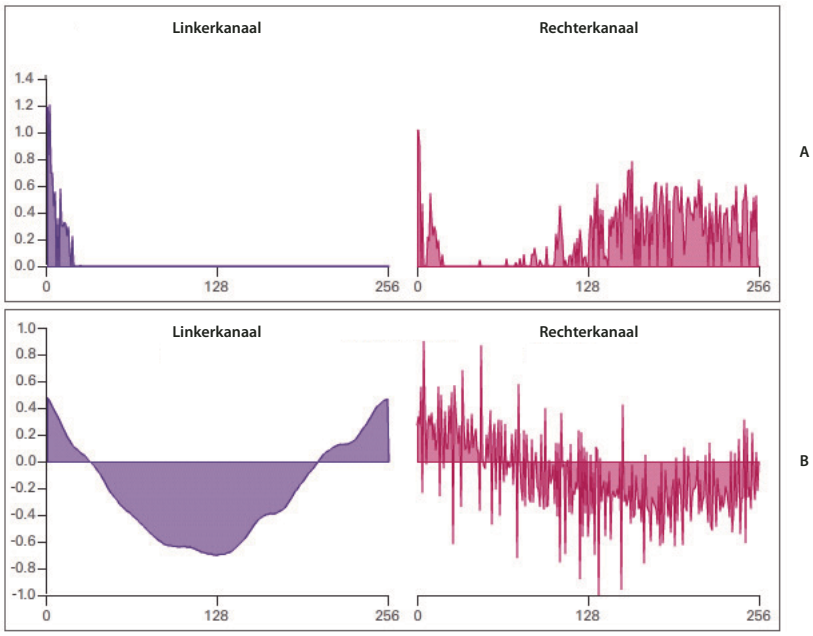
Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `SoundMixer.computeSpectrum()` wordt een toepassing in staat gesteld de onbewerkte geluidsgegevens voor de momenteel afgespeelde geluidsgolf te lezen. Als er momenteel meer dan één object SoundChannel wordt afgespeeld, worden met de methode `SoundMixer.computeSpectrum()` de gecombineerde geluidsgegevens van alle objecten SoundChannel samen weergegeven.

De geluidsgegevens worden geretourneerd als een object ByteArray met 512 bytes aan gegevens, die elk een zwevende-kommawaarde tussen -1 en 1 bevatten. Deze waarden vertegenwoordigen de amplitude van de punten in de geluidsvorm die wordt afgespeeld. De waarden worden in twee groepen van 256 aangeleverd: de eerste groep heeft betrekking op het linkerstereokanaal en de tweede groep heeft betrekking op het rechterstereokanaal.

De methode `SoundMixer.computeSpectrum()` retourneert frequentiespectrumgegevens in plaats van golfgegevens als de parameter `FFTMode` op `true` is ingesteld. Het frequentiespectrum laat amplitude gesorteerd op geluidsfrequentie zien, beginnen met de laatste frequentie. Er wordt een FFT-transformatie (Fast Fourier Transform) gebruikt om de golfgegevens in frequentiespectrumgegevens om te zetten. De resulterende frequentiespectrumwaarden variëren van 0 tot ongeveer 1,414 (de vierkantswortel van 2).

In het volgende diagram wordt een vergelijking gemaakt tussen enerzijds de door de methode `computeSpectrum()` geretourneerde gegevens waarvoor de parameter `FFTMode` is ingesteld op `true` en anderzijds de door deze methode geretourneerde gegevens waarvoor deze parameter is ingesteld op `false`. Het geluid waarvan de gegevens voor dit diagram zijn gebruikt, bevat een hard basgeluid in het linkerkanaal en een trommelgeluid in het rechterkanaal.



Waarden geretourneerd door methode SoundMixer.computeSpectrum()
***A.** fftMode = true **B.** fftMode = false*

De methode `computeSpectrum()` kan ook gegevens retourneren die opnieuw zijn bemonsterd met een lagere bitsnelheid. Dit leidt in het algemeen tot meer vloeiende golfgegevens of frequentiegegevens, wat ten koste gaat van het detailniveau. De parameter `stretchFactor` bestuurt de frequentie waarmee de methode `computeSpectrum()` gegevens bemonstert. Wanneer de parameter `stretchFactor` op 0 is ingesteld (de standaardwaarde), worden de geluidsgegevens bemonsterd met een frequentie van 44,1 kHz. De frequentie wordt bij elke volgende waarde van de parameter stretchFactor gehalveerd, wat betekent dat een waarde van 1 met een frequentie van 22,05 kHz correspondeert, dat een waarde van 2 met een frequentie van 11,025 kHz correspondeert, enzovoort. De methode `computeSpectrum()` blijft 256 bytes per stereokanaal retourneren wanneer er een hogere waarde voor `stretchFactor` wordt gebruikt.

De methode `SoundMixer.computeSpectrum()` kent een aantal beperkingen:

- Aangezien geluidsgegevens uit een microfoon of RTMP-streams het algemene object SoundMixer niet passeren, retourneert de methode `SoundMixer.computeSpectrum()` geen gegevens uit dergelijke bronnen.

- Als een of meer van de geluiden die worden afgespeeld, afkomstig zijn uit bronnen buiten de huidige inhoudssandbox, zorgen beveiligingsbeperkingen ervoor dat de methode `SoundMixer.computeSpectrum()` een fout genereert. Zie "Beveiligingsoverwegingen bij het laden en afspelen van geluiden" op pagina 478 en "Geladen media benaderen als gegevens" op pagina 1133 voor meer informatie over de beveiligingsbeperkingen van de methode `SoundMixer.computeSpectrum()`.

In een AIR-toepassing wordt de inhoud in de beveiligingssandbox van de toepassing (inhoud die met de AIR-toepassing is geïnstalleerd) echter niet beperkt door deze beveiligingsbeperkingen.

Een eenvoudig geluidsweergavesysteem maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

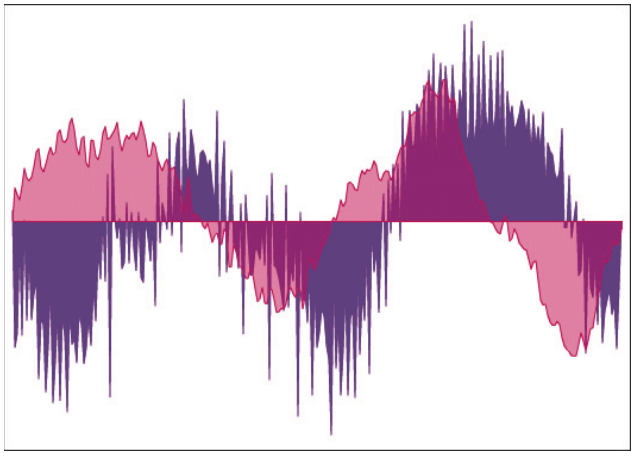
In het volgende voorbeeld wordt de methode `SoundMixer.computeSpectrum()` gebruikt om een grafiek van de geluidsgolf te tekenen die als animatie bij elk frame wordt weergegeven:

```
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);
}
```

In dit voorbeeld wordt eerst een geluidsbestand geladen en afgespeeld, waarna er wordt geluisterd naar de gebeurtenis `Event.ENTER_FRAME`, die ervoor zorgt dat de methode `onEnterFrame()` wordt geactiveerd terwijl het geluid wordt afgespeeld. De methode `onEnterFrame()` begint met het aanroepen van de methode `SoundMixer.computeSpectrum()`, die de geluidsgolfgegevens in het `bytes` object ByteArray opslaat.

De geluidsgolf wordt geplot via de API voor vectortekeningen. Een lus `for` doorloopt de eerste 256 gegevenswaarden, die met het linkerstereokanaal corresponderen, en tekent een lijn van elk punt naar het daarop volgende punt door gebruik te maken van de methode `Graphics.lineTo()`. Een tweede lus `for` doorloopt de volgende set van 256 waarden en zet deze in omgekeerde volgorde uit, van rechts naar links. De resulterende golfplots kunnen een interessant spiegelbeeldeffect opleveren, zoals in de volgende afbeelding wordt weergegeven.



Geluidsinvoer vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Via de klasse Microphone kan uw toepassing verbinding maken met een microfoon of een ander geluidsinvoerapparaat op het systeem van de gebruiker, en de invoeraudio vervolgens naar de luidsprekers van dat systeem verzenden of de audiogegevens naar een externe server verzenden, zoals Flash Media Server. U kunt de onbewerkte audiogegevens van de microfoon benaderen en deze opnemen of bewerken. U kunt de audio ook rechtstreeks naar de luidsprekers van het systeem sturen of gecomprimeerde audiogegevens naar een externe server zenden. U kunt de codec Speex of Nellymoser gebruiken voor gegevens die worden verzonden naar een externe server. (De Speex-codec wordt ondersteund vanaf Flash Player 10 en Adobe AIR 1.5.)

Meer Help-onderwerpen

Michael Chaize: AIR, Android, and the Microphone

Christophe Coenraets: Voice Notes for Android

Een microfoon gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Microphone heeft geen constructormethode. In plaats daarvan gebruikt u de statische methode `Microphone.getMicrophone()` om een nieuwe instantie Microphone te verkrijgen, zoals hieronder wordt weergegeven:

```
var mic:Microphone = Microphone.getMicrophone();
```

Als de methode `Microphone.getMicrophone()` zonder parameter wordt aangeroepen, wordt het eerste geluidsinvoerapparaat geretourneerd dat op het systeem van de gebruiker wordt gedetecteerd.

Er kan meer dan één geluidsinvoerapparaat op een systeem aangesloten zijn. Uw toepassing kan de eigenschap `Microphone.names` gebruiken om een array met de namen van alle beschikbare geluidsinvoerapparaten op te vragen. Vervolgens kan de toepassing de methode `Microphone.getMicrophone()` aanroepen, waarbij de parameter `index` overeenkomt met de indexwaarde van de naam van een apparaat in de array.

Het is mogelijk dat er geen microfoon of ander geluidsinvoerapparaat op een systeem is aangesloten. U kunt de eigenschap `Microphone.names` of de methode `Microphone.getMicrophone()` gebruiken om na te gaan of de gebruiker een geluidsinvoerapparaat heeft geïnstalleerd. Als de gebruiker geen geluidsinvoerapparaat heeft geïnstalleerd, heeft de array `names` een lengte van nul, en retourneert de methode `getMicrophone()` de waarde `null`.

Wanneer uw toepassing de methode `Microphone.getMicrophone()` aanroept, wordt in Flash Player het dialoogvenster Flash Player Settings weergegeven, waarin de gebruiker wordt gevraagd om Flash Player al dan niet toegang tot de camera en microfoon op het systeem te verlenen. Nadat de gebruiker in dit dialoogvenster op de knop Allow of de knop Deny heeft geklikt, wordt de gebeurtenis StatusEvent verzonden. De eigenschap `code` van die StatusEvent-instantie geeft aan of microfoontoegang wordt toegestaan dan wel geblokkeerd, zoals wordt weergegeven in dit voorbeeld:

```
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.");
    }
}
```

De eigenschap `StatusEvent.code` bevat de waarde Microphone.Unmuted als er toegang wordt verleend, en Microphone.Muted als toegang wordt geweigerd.

De eigenschap `Microphone.muted` wordt respectievelijk op `true` of `false` ingesteld wanneer de gebruiker microfoontoegang toestaat of blokkeert. De eigenschap `muted` wordt echter pas voor de instantie Microphone ingesteld nadat de gebeurtenis StatusEvent is verzonden, zodat uw toepassing ook moet wachten tot de gebeurtenis `StatusEvent.STATUS` is verzonden alvorens de waarde van de eigenschap `Microphone.muted` op te vragen.

Flash Player kan het instellingenvenster alleen weergeven als het toepassingsvenster groot genoeg is (minimaal 215 x 138 pixels). Anders wordt de toegang automatisch geweigerd.

Voor in de AIR-toepassingssandbox uitgevoerde inhoud is geen toestemming van de gebruiker nodig om de microfoon te benaderen. Er worden dus nooit statusgebeurtenissen voor het dempen en dempen ongedaan maken van de microfoon verzonden. Voor in AIR buiten de toepassingssandbox uitgevoerde inhoud is wel toestemming van de gebruiker nodig en deze statusgebeurtenissen kunnen dus worden verzonden.

Microfoonaudio naar lokale luidsprekers leiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Audio-invoer van een microfoon kan naar de lokale systeemluidsprekers worden geleid door de methode `Microphone.setLoopback()` met parameterwaarde `true` aan te roepen.

Wanneer geluid uit een lokale microfoon naar lokale luidsprekers wordt geleid, zou een audiofeedbacklus kunnen ontstaan, waardoor harde piepgeluiden worden afgegeven en de geluidshardware beschadigd kan raken. Als de methode `Microphone.setUseEchoSuppression()` met parameterwaarde `true` wordt aangeroepen, wordt de kans op audiofeedback kleiner, maar blijft aanwezig. Het wordt aanbevolen om `Microphone.setUseEchoSuppression(true)` altijd aan te roepen voordat `Microphone.setLoopback(true)` wordt aangeroepen, tenzij u er zeker van bent dat de gebruiker het geluid via een hoofdtelefoon of andere apparatuur dan luidsprekers afspeelt.

In de volgende code ziet u hoe u de audio van een lokale microfoon naar de lokale systeemluidsprekers kunt leiden:

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

Microfoonaudio aanpassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassing kan de audiogegevens die uit een microfoon afkomstig zijn, op twee manieren aanpassen. Ten eerste kan de versterking van het invoergeluid worden gewijzigd, waarbij in feite de invoerwaarden met een opgegeven factor worden vermenigvuldigd om een harder of zachter geluid te bereiken. De eigenschap `Microphone.gain` accepteert numerieke waarden van 0 tot en met 100. De waarde 50 fungeert als vermenigvuldigingsfactor 1, waarmee het basisvolume wordt opgegeven. De waarde 0 fungeert als vermenigvuldigingsfactor 0, waarmee de invoeraudio volledig wordt gedempt. Bij waarden van meer dan 50 wordt het volume verhoogd.

Uw toepassing kan ook de bemonsteringsfrequentie van de invoeraudio wijzigen. Hogere bemonsteringsfrequenties corresponderen met hogere geluidskwaliteit, maar leiden ook tot zwaardere gegevensstreams die meer resources voor overdracht en opslag gebruiken. De eigenschap `Microphone.rate` vertegenwoordigt de audiobemonsteringsfrequentie gemeten in kilohertz (kHz). De standaardbemonsteringsfrequentie is 8 kHz. U kunt de eigenschap `Microphone.rate` op een hogere waarde dan 8 kHz instellen als uw microfoon de hogere frequentie ondersteunt. Als de eigenschap `Microphone.rate` bijvoorbeeld op 11 wordt ingesteld, wordt de samplingfrequentie ingesteld op 11 kHz; bij een waarde van 22 wordt de samplingfrequentie ingesteld op 22 kHz enzovoort. Welke samplingfrequenties beschikbaar zijn, is afhankelijk van de geselecteerde codec. Wanneer u de codec Nellymoser gebruikt, kunt u 5, 8, 11, 16, 22 en 44 kHz opgeven als samplingfrequentie. Wanneer u Speex-codec gebruikt (beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5), bent u beperkt tot 16 kHz.

Microfoonactiviteit detecteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Om bandbreedte en verwerkingsresources te besparen, probeert Flash Player na te gaan wanneer er geen geluid door een microfoon wordt overgebracht. Wanneer het activiteitsniveau van de microfoon gedurende een bepaalde tijdsperiode onder het stiltedrempelniveau blijft, stopt Flash Player met de overdracht van audio-invoer en wordt in plaats daarvan de eenvoudige gebeurtenis ActivityEvent verzonden. Als u de Speex-codec gebruikt (beschikbaar in Flash Player 10 of hoger en Adobe AIR 1.5 of hoger), stelt u het stilteniveau in op 0 om ervoor te zorgen dat de toepassing continu audiogegevens overzendt. Bij de detectie van Speex-spraakactiviteit wordt de bandbreedte automatisch gereduceerd.

Opmerking: *Een Microphone-object verzendt alleen Activity-gebeurtenissen wanneer uw toepassing de microfoon controleert. Als u dus niet* `setLoopBack( true )` *aanroept, een listener toevoegt voor voorbeeldgegevensgebeurtenissen of de microfoon koppelt aan een NetStream-object, worden er geen Activity-gebeurtenissen verzonden.*

Drie eigenschappen van de klasse Microphone volgen en controleren de activiteitsdetectie:

- De alleen-lezen-eigenschap `activityLevel` vertegenwoordigt de hoeveelheid geluid die door de microfoon wordt gedetecteerd, op een schaal van 0 tot 100.
- Via de eigenschap `silenceLevel` wordt opgegeven hoeveel geluid nodig is om de microfoon te activeren en de gebeurtenis `ActivityEvent.ACTIVITY` te verzenden. Ook voor de eigenschap `silenceLevel` wordt een schaal van 0 tot 100 gebruikt; de standaardwaarde is 10.
- Met de eigenschap `silenceTimeout` wordt opgegeven gedurende hoeveel milliseconden het activiteitsniveau onder het stilteniveau moet blijven voordat de gebeurtenis `ActivityEvent.ACTIVITY` wordt verzonden ter indicatie dat er nu geen geluid uit de microfoon komt. De standaardwaarde voor `silenceTimeout` is 2000.

Zowel de eigenschap `Microphone.silenceLevel` als de eigenschap `Microphone.silenceTimeout` is alleen-lezen, maar de waarden van deze eigenschappen kunnen worden gewijzigd via de methode `Microphone.setSilenceLevel()`.

In sommige gevallen kan het proces ter activering van de microfoon bij detectie van nieuwe activiteit tot een korte vertraging leiden. Dergelijke activeringsvertragingen kunnen worden voorkomen door de microfoon altijd actief te houden. Uw toepassing kan de methode `Microphone.setSilenceLevel()` met de parameter `silenceLevel` ingesteld op nul gebruiken om Flash Player de opdracht te geven om de microfoon actief te houden, zelfs als er geen geluid wordt gedetecteerd. Als de parameter `silenceLevel` daarentegen op 100 wordt ingesteld, kan de microfoon helemaal niet worden geactiveerd.

In het volgende voorbeeld wordt informatie over de microfoon weergegeven en wordt melding gemaakt van activiteitsgebeurtenissen en statusgebeurtenissen die door een object Microphone worden verzonden:

```
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);
}
```

Wanneer u het bovenstaande voorbeeld uitvoert, moet u spreken of andere geluiden maken in uw systeemmicrofoon en de instructies trace bekijken die worden weergegeven in een console- of foutopsporingsvenster.

Audio naar en van een mediaserver verzenden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er zijn aanvullende audiomogelijkheden beschikbaar bij gebruik van ActionScript met een streaming-mediaserver zoals Flash Media Server.

Zo kan uw toepassing bijvoorbeeld een object Microphone aan een object NetStream koppelen en gegevens rechtstreeks van de microfoon van de gebruiker naar de server verzenden. Audiogegevens kunnen ook van de server worden gestreamd naar een toepassing die is gemaakt met Flash of Flex, en worden afgespeeld als onderdeel van een filmfragment of door gebruik te maken van een Video-object.

De Speex-codec is beschikbaar vanaf Flash Player 10 en Adobe AIR 1.5. Stel de eigenschap `codec` van het Microphone-object in om de codec in te stellen die wordt gebruikt voor gecomprimeerde audio die naar de mediaserver is verzonden. Deze eigenschap kan twee waarden hebben die worden opgesomd in de klasse SoundCodec. Stel de eigenschap codec in op `SoundCodec.SPEEX` als u de codec Speex wilt selecteren voor de compressie van geluid. Stel de eigenschap codec in op `SoundCodec.NELLYMOSER` (standaardwaarde) als u de codec Nellymoser wilt selecteren voor de compressie van geluid.

Zie de onlinedocumentatie bij Flash Media Server op www.adobe.com/go/learn_fms_docs_nl voor meer informatie.

Geluidsgegevens van een microfoon vastleggen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

In Flash Player 10.1. en AIR 2 of later kunt u geluidsgegevens van een microfoon vastleggen als een bytearray of als zwevende-kommawaarden. Elke waarde vertegenwoordigt een sample van homofone audiogegevens.

Als u microfoongegevens wilt opvragen, stel u een gebeurtenislistener in voor de gebeurtenis `sampleData` van het Microphone-object. Het Microphone-object verzendt periodiek `sampleData`-gebeurtenissen, terwijl de microfoonbuffer wordt gevuld met geluidssamples. Het SampleDataEvent-object heeft een `data`-eigenschap die bestaat uit een bytearray van geluidssamples. Deze samples worden allemaal vertegenwoordigd als zwevende-kommawaarden, die elk een homofoon geluidssample zijn.

In de volgende code worden geluidsgegevens van een microfoon vastgelegd in een ByteArray-object met de naam `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);
    }
}
```

U kunt de samplebytes hergebruiken als audio die kan worden afgespeeld voor een Sound-object. Als u dit doet, moet u de eigenschap `rate` van het Microphone-object op 44 instellen. Dit is de samplingfrequentie die wordt gebruikt door Sound-objecten. (Microfoonsamples die zijn vastgesteld met een frequentie die lager ligt dan de 44 kHz die wordt vereist door het Sound-object, kunnen worden geconverteerd.) Denk er ook aan dat het Microphone-object homofone samples vastlegt, terwijl het Sound-object stereogeluid gebruikt. Dit betekent dat u elke byte die door het Microphone-object wordt vastgelegd, twee keer naar het Sound-object moet schrijven. In het volgende voorbeeld worden 4 seconden aan geluidsgegevens van een microfoon vastgelegd en worden deze gegevens met een Sound-object afgespeeld:

```
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.");
}
```

Zie “Werken met dynamisch gegenereerde audio” op pagina 472 voor meer informatie over het afspelen van geluiden van geluidssamplegegevens.

Voorbeeld van geluid: Podcast-speler

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een podcast is een geluidsbestand dat via internet wordt gedistribueerd, op aanvraag of in het kader van een abonnement. Podcasts worden meestal gepubliceerd als onderdeel van een reeks, ook wel podcast-kanaal genoemd. Aangezien podcast-afleveringen één minuut, maar ook meerdere uren kunnen duren, worden deze meestal gestreamd tijdens het afspelen. Podcast-afleveringen, ook wel items genoemd, worden meestal aangeleverd in MP3-indeling. Video-podcasts zijn ook populair, maar in deze voorbeeldtoepassing worden alleen audio-podcasts afgespeeld waarvoor MP3-bestanden worden gebruikt.

Dit voorbeeld is geen aggregatortoepassing met volledige podcast-functionaliteit. De voorbeeldtoepassing voorziet bijvoorbeeld niet in het beheer van abonnementen op specifieke podcasts en houdt niet bij welke podcasts de gebruiker al heeft beluisterd, zodat deze informatie niet beschikbaar is als de gebruiker de toepassing opnieuw uitvoert. De voorbeeldtoepassing is een goed uitgangspunt voor een aggregator met volledige podcast-functionaliteit.

In het voorbeeld van de podcast-speler worden de volgende ActionScript-programmeringstechnieken geïllustreerd:

- Een externe RSS-feed lezen en de XML-inhoud van de feed parseren
- Een klasse SoundFacade maken om het laden en afspelen van geluidsbestanden te vereenvoudigen
- De voortgang van het afspelen van geluid weergeven
- Het afspelen van geluid onderbreken en hervatten

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De bestanden voor de toepassing Podcast Player zijn te vinden in de map Samples/PodcastPlayer. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
PodcastPlayer.mxml of PodcastPlayer.fla	De gebruikersinterface voor de toepassing voor Flex (MXML) of Flash (FLA).
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	Klasse Document die de gebruikersinterfacelogica bevat voor de podcast-player (alleen Flash).
SoundPlayer.mxml	Een MXML-component die afspeelknoppen en voortgangsbalken weergeeft en die het afspelen van geluid regelt (alleen voor Flex).
main.css	Stijlen voor de gebruikersinterface van de toepassing (alleen Flex).
afbeeldingen/	Pictogrammen voor het opmaken van de knoppen (alleen Flex).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	Klasse voor het filmclipsymbool SoundPlayer dat de gebruikersinterfacelogica bevat voor de geluidspeler (alleen Flash).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	Aangepaste celrenderer voor het weergeven van een knop Afspelen in een cel van een gegevensraster (alleen Flash).
com/example/programmingas3/podcastplayer/RSSBase.as	Een basisklasse die algemene eigenschappen en methoden voor de klassen RSSChannel en RSSItem biedt.

Bestand	Beschrijving
com/example/programmingas3/podcastplayer/RSSChannel.as	Een ActionScript-klasse die gegevens over een RSS-kanaal bevat.
com/example/programmingas3/podcastplayer/RSSItem.as	Een ActionScript-klasse die gegevens over een RSS-item bevat.
com/example/programmingas3/podcastplayer/SoundFacade.as	De belangrijkste ActionScript-klasse voor de toepassing. Deze omvat de methoden en gebeurtenissen van de klassen Sound en SoundChannel, aangevuld met ondersteuning voor het onderbreken en hervatten van het afspelen.
com/example/programmingas3/podcastplayer/URLService.as	Een ActionScript-klasse die gegevens van een externe URL ophaalt.
playerconfig.xml	Een XML-bestand met een lijst van RSS-feeds voor podcast-kanalen.
comp/example/programmingas3/utils/DateUtil.as	Klasse die wordt gebruikt om datums gemakkelijk op te maken (alleen Flash).

RSS-gegevens voor een podcast-kanaal lezen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing Podcast Player begint met het lezen van informatie over een aantal podcast-kanalen en corresponderende afleveringen:

1. Eerst leest de toepassing een XML-configuratiebestand met een lijst van podcast-kanalen en geeft een lijst met kanalen voor de gebruiker weer.

2. Wanneer de gebruiker een van de podcast-kanalen selecteert, wordt de RSS-feed voor het kanaal gelezen en wordt een lijst met kanaalafleveringen weergegeven.

In dit voorbeeld wordt de hulpprogrammaklasse URLLoader gebruikt om tekstgegevens van een externe locatie of uit een lokaal bestand op te halen. De Podcast Player maakt eerst een object URLLoader om een lijst met RSS-feeds in XML-indeling uit het bestand playerconfig.xml op te halen. Wanneer de gebruiker vervolgens een specifieke feed in de lijst selecteert, wordt een nieuw object URLLoader gemaakt om de RSS-gegevens uit de URL van die feed te lezen.

Het laden en afspelen van geluid vereenvoudigen met behulp van de klasse SoundFacade

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De ActionScript 3.0-geluidsarchitectuur is krachtig, maar complex. In toepassingen waarvoor alleen elementaire functies voor het laden en afspelen en geluiden nodig zijn, kan een klasse worden gebruikt waarmee de complexiteit deels wordt verborgen door een vereenvoudigde set methodeaanroepen en gebeurtenissen te bieden. In de wereld van softwareontwerppatronen wordt een dergelijke klasse een *facade* genoemd.

De klasse SoundFacade biedt één enkele interface voor het uitvoeren van de volgende taken:

- Geluidsbestanden laden met behulp van een object Sound, een object SoundLoaderContext en de klasse SoundMixer
- Geluidsbestanden afspelen met behulp van het object Sound en het object SoundChannel

- Gebeurtenissen over de voortgang van het afspelen verzenden
- Het afspelen van het geluid onderbreken en hervatten met behulp van het object Sound en het object SoundChannel

De klasse SoundFacade probeert het merendeel van de functionaliteit van de ActionScript-geluidsklassen te bieden en daarbij de complexiteit beperkt te houden.

In de volgende code worden de klassendeclaratie, de klasseneigenschappen en de constructormethode `SoundFacade()` weergegeven:

```
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();
        }
    }
```

De klasse SoundFacade vormt een uitbreiding van de klasse EventDispatcher, zodat deze eigen gebeurtenissen kan verzenden. In de klassencode worden eerst eigenschappen voor een object Sound en een object SoundChannel gedeclareerd. In de klasse wordt ook de waarde van de URL van het geluidsbestand opgeslagen, evenals de eigenschap `bufferTime` voor gebruik bij streaming van het geluid. Bovendien worden er bepaalde Booleaanse parameterwaarden geaccepteerd, die invloed hebben op het laad- en afspeelgedrag:

- Via de parameter `autoLoad` krijgt het object de opdracht om met het laden van het geluid te beginnen zodra dit object is gemaakt.
- Met de parameter `autoPlay` wordt aangegeven dat het afspelen van het geluid moet starten zodra er voldoende geluidsgegevens zijn geladen. Als dit streaming geluid betreft, wordt het afspelen gestart zodra er voldoende gegevens, zoals opgegeven door de eigenschap `bufferTime`, zijn geladen.
- Met de parameter `streaming` wordt aangegeven dat er met het afspelen van dit geluidsbestand kan worden begonnen voordat het laden is voltooid.

De parameter `bufferTime` is standaard ingesteld op -1. Als de constructormethode een negatieve waarde in de parameter `bufferTime` detecteert, wordt de eigenschap `bufferTime` ingesteld op de waarde van `SoundMixer.bufferTime`. Zo kan de toepassing desgewenst standaard worden ingesteld op de algemene waarde van `SoundMixer.bufferTime`.

Als de parameter `autoLoad` op `true` is ingesteld, roept de constructormethode onmiddellijk de volgende methode `load()` aan om te beginnen met het laden van het geluidsbestand:

```
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);
}
```

Met de methode `load()` wordt een nieuw object Sound gemaakt, waarna listeners voor alle belangrijke geluidsgebeurtenissen worden toegevoegd. Vervolgens krijgt het object Sound de opdracht het geluidsbestand te laden, waarbij een object SoundLoaderContext wordt gebruikt om de waarde voor `bufferTime` door te geven.

Aangezien de eigenschap `url` kan worden gewijzigd, kan er een instantie SoundFacade worden gebruikt om verschillende geluidsbestanden na elkaar af te spelen: wijzig eenvoudig de eigenschap `url` en roep de methode `load()` aan; het nieuwe geluidsbestand wordt geladen.

De volgende drie methoden voor gebeurtenislisteners laten zien hoe het object SoundFacade de voortgang van het laden traceert en bepaalt wanneer er wordt gestart met het afspelen van het geluid:

```
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();
    }
}
```

De methode `onLoadOpen()` wordt uitgevoerd zodra het laden van het geluid wordt gestart. Als het geluid in streaming modus kan worden afgespeeld, stelt de methode `onLoadComplete()` de markering `isReadyToPlay` direct in op `true`. De markering `isReadyToPlay` bepaalt of de toepassing met het afspelen van het geluid kan beginnen, bijvoorbeeld na een bepaalde gebruikershandeling, zoals het klikken op een afspeelknop. De klasse SoundChannel beheert de buffering van geluidsgegevens, wat betekent dat het niet nodig is om expliciet te controleren of er voldoende gegevens zijn geladen voordat de methode `play()` wordt aangeroepen.

De methode `onLoadProgress()` wordt periodiek aangeroepen tijdens het laadproces. Er wordt een kloon van het object ProgressEvent verzonden, die kan worden gebruikt door code die gebruikmaakt van dit object SoundFacade.

Wanneer de geluidsgegevens volledig zijn geladen, wordt de methode `onLoadComplete()` uitgevoerd en wordt zo nodig de methode `play()` voor niet-streaming geluiden aangeroepen. De methode `play()` zelf wordt hieronder weergegeven.

```
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();
        }
    }
}
```

De methode `play()` roept de methode `Sound.play()` aan als het geluid gereed is om te worden afgespeeld. Het resulterende object SoundChannel wordt opgeslagen in de eigenschap `sc`. De methode `play()` maakt vervolgens een object Timer dat wordt gebruikt om gebeurtenissen over de voortgang van het afspelen met een regelmatig interval te verzenden.

De voortgang van het afspelen weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het maken van een object Timer ter controle van het afspelen is een complexe aangelegenheid, waarvoor u slechts één keer code zou moeten schrijven. Door deze Timer-logica onder te brengen in een herbruikbare klasse zoals de klasse SoundFacade stelt u toepassingen in staat naar dezelfde soorten voortgangsgebeurtenissen te luisteren wanneer een geluid wordt geladen als wanneer het wordt afgespeeld.

Het object Timer dat door de methode `SoundFacade.play()` wordt gemaakt, verzendt elke seconde een instantie TimerEvent. De volgende methode `onPlayTimer()` wordt uitgevoerd zodra er een nieuwe TimerEvent arriveert:

```
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);
}
```

De methode `onPlayTimer()` implementeert de techniek voor het schatten van grootte die wordt beschreven in de sectie "De status van het afspelen volgen" op pagina 476. Vervolgens wordt er een nieuwe instantie ProgressEvent met het gebeurtenistype `SoundFacade.PLAY_PROGRESS` gemaakt, waarbij de eigenschap `bytesLoaded` is ingesteld op de huidige positie van het object SoundChannel en de eigenschap `bytesTotal` is ingesteld op de geschatte lengte van de geluidsgegevens.

Afspelen onderbreken en hervatten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eerder weergegeven methode `SoundFacade.play()` accepteert de parameter `pos`, die correspondeert met een startpositie in de geluidsgegevens. Als de waarde van `pos` nul is, wordt het afspelen van het geluid vanaf het begin gestart.

De methode `SoundFacade.stop()` accepteert ook de parameter `pos` zoals deze hier wordt weergegeven:

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

Steeds wanneer de methode `SoundFacade.stop()` wordt aangeroepen, wordt de eigenschap `pausePosition` zodanig ingesteld dat de toepassing weet waar de afspeelkop moet worden geplaatst als de gebruiker het afspelen van datzelfde geluid wil hervatten.

De hieronder weergegeven methoden `SoundFacade.pause()` en `SoundFacade.resume()` roepen respectievelijk de methode `SoundFacade.stop()` en `SoundFacade.play()` aan, waarbij steeds een waarde voor de parameter `pos` wordt doorgegeven.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

De methode `pause()` geeft de huidige waarde van `SoundChannel.position` door aan de methode `play()`, die deze waarde in de eigenschap `pausePosition` opslaat. De methode `resume()` begint opnieuw met het afspelen van hetzelfde geluid, waarbij de waarde van `pausePosition` als startpunt wordt gebruikt.

Het Podcast Player-voorbeeld uitbreiden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In dit voorbeeld wordt een elementaire podcast-speler gepresenteerd, waarmee het gebruik van de herbruikbare klasse SoundFacade wordt geïllustreerd. U kunt andere functies toevoegen om het nut van deze toepassing verder uit te breiden, zoals:

- De lijst van feeds en gebruiksgegevens over elke aflevering opslaan in een instantie SharedObject die kan worden gebruikt wanneer de gebruiker de toepassing weer gebruikt.
- De gebruiker toestaan eigen RSS-feeds aan de lijst met podcast-kanalen toe te voegen.

- De positie van de afspeelkop onthouden wanneer de gebruiker een aflevering stopzet of verlaat, zodat deze vanaf dat punt kan worden hervat wanneer de gebruiker de toepassing opnieuw uitvoert.
- MP3-bestanden met afleveringen downloaden om deze offline te kunnen beluisteren, wanneer de gebruiker niet met internet is verbonden.
- Abonnementsfuncties toevoegen waarmee periodiek op nieuwe afleveringen in een podcast-kanaal wordt gecontroleerd en de lijst met afleveringen automatisch wordt bijgewerkt.
- Zoek- en bladerfunctionaliteit voor podcasts toevoegen met behulp van een API van een podcast-hostingservice zoals Odeo.com.

Hoofdstuk 25: Werken met video

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Video is een van de vooraanstaande technologieën op internet. De traditionele videopresentatie (in een rechthoekig scherm met een voortgangsbalk en bedieningsknoppen eronder) is echter slechts één manier om video te gebruiken. Met ActionScript hebt u fijngeregelde toegang tot en controle over het laden, presenteren en afspelen van video.

Basisbeginselen van video

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een van de belangrijkste mogelijkheden van Adobe® Flash® Player en Adobe® AIR™ is het weergeven en manipuleren van videogegevens met ActionScript. Dit gebeurt op dezelfde manier als het manipuleren van andere visuele inhoud, zoals afbeeldingen, animaties, tekst, enzovoort. Wanneer u een FLV-bestand (Flash Video) in Adobe Flash CS4 Professional maakt, kunt u een skin met algemene afspeelbesturingselementen selecteren. U hoeft zich echter niet tot de beschikbare opties te beperken. Wanneer u ActionScript gebruikt, kunt u de besturing van het laden, weergeven en afspelen van video nauwkeurig instellen (u kunt uw eigen skin voor de videospeler maken of uw video op een minder traditionele manier gebruiken). Wanneer u met video werkt in ActionScript, gebruikt u een combinatie van verschillende klassen:

- Klasse Video: het klassieke tekstvak met video-inhoud in het werkgebied is een instantie van de klasse Video. De klasse Video is een weergaveobject dat met dezelfde technieken kan worden gemanipuleerd als andere weergaveobjecten, zoals plaatsing, transformaties, filters, overvloeimodi, enzovoort.
- Klasse StageVideo: de Video-klasse gebruikt doorgaans softwaredecodering en -rendering. Wanneer GPU-hardwareversnelling beschikbaar is op een apparaat, kan uw toepassing presentatie met hardwareversnelling optimaal benutten door over te schakelen op de klasse StageVideo. De StageVideo-API bevat een serie gebeurtenissen die uw code melden wanneer geschakeld moet worden tussen StageVideo- en Video-objecten. Er gelden enkele kleine beperkingen op het afspelen van werkgebiedvideo. Implementeer de StageVideo-API als uw toepassing deze beperkingen accepteert. Zie “Richtlijnen en beperkingen” op pagina 541.
- Klasse Netstream: wanneer u een videobestand laadt dat door ActionScript wordt bestuurd, vertegenwoordigt een NetStream-instantie de bron van de video-inhoud (in dit geval een stream van videogegevens). Bij een NetStream-instantie wordt ook een NetConnection-object gebruikt. Dit regelt de verbinding met het videobestand (zoals de tunnel waardoor de videogegevens worden geleid).
- Klasse Camera: wanneer u videogegevens gebruikt van een camera die aan de computer van de gebruiker is gekoppeld, vertegenwoordigt een Camera-instantie de bron van de video-inhoud (de camera van de gebruiker en de videogegevens die beschikbaar worden gemaakt). U kunt een camera gebruiken om StageVideo te voeden. Deze functie is nieuw in Flash Player 11.4 en AIR 3.4.

Wanneer u externe video laadt, kunt u het bestand van een standaardwebserver laden om af te spelen met progressieve download. U kunt ook met streamingvideo werken die door een gespecialiseerde server wordt geleverd, zoals Flash® Media Server van Adobe.

Belangrijke concepten en termen

Actiepunt Een markering die op een bepaald tijdstip in een videobestand kan worden geplaatst. Deze werkt bijvoorbeeld als bladwijzer om de positie van dat tijdstip te bepalen of om aanvullende gegevens te verschaffen die aan dat tijdstip zijn gekoppeld.

Codering Het proces waarbij videogegevens van de ene naar de andere indeling worden omgezet; bijvoorbeeld een videobron met een hoge resolutie die wordt omgezet naar een indeling die geschikt is voor distributie via internet.

Frame Eén segment videogegevens. Elk frame is een soort stilstaande afbeelding die een momentopname op een bepaald tijdstip vertegenwoordigt. Wanneer u frames op hoge snelheid achter elkaar afspeelt, wordt de illusie van beweging gecreëerd.

Hoofdframe Een videoframe dat de volledige gegevens voor het frame bevat. Andere frames die op een hoofdframe volgen, bevatten alleen informatie over hoe ze verschillen van het hoofdframe, in plaats van dat ze de volledige informatiewaarde van het frame bevatten.

Metagegevens Informatie over een videobestand die is ingesloten in het videobestand en wordt opgehaald wanneer de video is geladen.

Progressieve download Wanneer een videobestand vanaf een standaardwebserver wordt geleverd, worden videogegevens met progressieve download geladen. Dit houdt in dat de videogegevens op volgorde worden geladen. Dit heeft als voordeel dat de video kan beginnen met afspelen voordat het gehele bestand is gedownload. U kunt echter niet naar een deel van de video springen dat nog niet is geladen.

Streaming Als alternatief voor progressieve download kunt u een speciale videoserver gebruiken om video te leveren via internet. Deze techniek wordt streaming (of true streaming) genoemd. Bij streaming wordt het gehele videobestand nooit in één keer naar de computer van de kijker gedownload. De computer heeft steeds slechts een deel van de totale videogegevens nodig. Hierdoor wordt het downloaden versneld. Aangezien de levering van de video-inhoud door een speciale server wordt bestuurd, kunt u alle delen van de video op elk gewenst moment bekijken. U hoeft dus niet meer te wachten tot de volledige video is gedownload voordat u deze kunt openen.

Video-indelingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naast de video-indeling FLV van Adobe ondersteunen Flash Player en Adobe AIR video en audio die is gecodeerd in de bestandsindelingen H.264 en HE-AAC conform de norm MPEG-4. Deze indelingen streamen video van hoge kwaliteit bij lagere bitsnelheden. Ontwikkelaars kunnen met industriestandaardsoftware, zoals Adobe Premiere Pro en Adobe After Effects, aantrekkelijke video-inhoud maken en leveren.

Type	Indeling	Container
Video	H.264	MPEG-4: MP4, M4V, F4V, 3GPP
Video	Sorenson Spark	FLV-bestand
Video	ON2 VP6	FLV-bestand
Audio	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4:MP4, M4V, F4V, 3GPP
Audio	MP3	MP3
Audio	Nellymoser	FLV-bestand
Audio	Speex	FLV-bestand

Meer Help-onderwerpen

Flash Media Server: ondersteunde codecs

Adobe dynamische streaming via HTTP

Video coderen voor mobiele apparatuur

AIR op Android kan een uitgebreid scala van H.264-video's decoderen. Slechts een klein aantal H.264-video's is echter geschikt om probleemloos te worden afgespeeld op mobiele telefoons. Dat is te wijten aan het beperkte verwerkingsvermogen van vele mobiele telefoons. Adobe Flash Player voor mobiele apparatuur kan H.264-video's decoderen aan de hand van geïntegreerde hardwareversnelling. Deze manier van decoderen leidt tot hogere kwaliteit bij een lager energieverbruik.

De H.264-standaard ondersteunt meerdere coderingstechnieken. Alleen op geavanceerde apparaten kunnen video's met complexe profielen en niveaus probleemloos worden afgespeeld. In basislijnprofiel gecodeerde video kan echter op de meeste apparaten worden afgespeeld. Op mobiele apparaten is hardwareversnelling beschikbaar voor een subset van deze technieken. De profiel- en niveauparameters definiëren deze subset coderingstechnieken en instellingen die door de codeermodule worden gebruikt. Voor ontwikkelaars betekent dit het coderen van video in de geselecteerde resolutie die op de meeste apparaten goed wordt afgespeeld.

Hoewel de resolutie die baat heeft bij hardwareversnelling per apparaat varieert, ondersteunen de meeste apparaten de volgende standaardresoluties.

Verhouding	**Aanbevolen resoluties**		
4:3	640 × 480	512 × 384	480 × 360
16:9	640 × 360	512 x 288	480 × 272

Opmerking: *Flash Player ondersteunt elk niveau en elk profiel van de H.264-standaard. Houd u aan deze aanbevelingen voor hardwareversnelling en betere ervaringen op de meeste apparaten. Deze aanbevelingen zijn niet verplicht.*

Zie Aanbevelingen voor het coderen van H.264-video voor Flash Player 10.1 op mobiele apparaten voor een gedetailleerde discussie en coderingsinstellingen in Adobe Media Encoder CS5.

Opmerking: *Op iOS kan alleen met de Sorenson Spark- en On2 VP6-codecs gecodeerde video worden afgespeeld met gebruik van de klasse Video. U kunt via H.264 gecodeerde video in het videoafspeelapparaat afspelen door de URL naar de video te starten aan de hand van de functie* `flash.net.navigateToURL()`*. U kunt H.264-video ook afspelen aan de hand van de <video>-tag in een HTML-pagina die wordt weergegeven in een StageWebView-object.*

Flash Player- en AIR-compatibiliteit met gecodeerde videobestanden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player 7 ondersteunt FLV-bestanden die zijn gecodeerd met de video-codec van Sorenson™ Spark™. Flash Player 8 ondersteunt FLV-bestanden die zijn gecodeerd met het coderingsprogramma van Sorenson Spark of On2 VP6 in Flash Professional 8. De video-codec van On2 VP6 ondersteunt een alfakanaal.

Flash Player 9.0.115.0 en hoger ondersteunen bestanden die zijn afgeleid van de standaard MPEG-4 containerindeling. Zoals F4V, MP4, M4A, MOV, MP4V, 3GP en 3G2, op voorwaarde dat deze H.264-video, met HE-AAC v2 gecodeerde audio of allebei bevatten. H.264 levert video van betere kwaliteit bij lagere bitsnelheden ten opzichte van hetzelfde coderingsprofiel in Sorenson of On2. HE-AAC v2 is een uitbreiding van AAC, een standaardgeluidsindeling die is gedefinieerd in de MPEG-4 videostandaard. HE-AAC v2 gebruikt SBR- (Spectral Band Replication) en PS-technieken (Parametric Stereo) om de codering en bitsnelheden te optimaliseren.

In de volgende tabel vindt u de ondersteunde codecs. U ziet hier ook de corresponderende SWF-bestandsindeling en de versies van Flash Player en AIR die vereist zijn om ze af te spelen:

Codec	**Versie SWF-bestandsindeling (laagste ondersteunde publicatieversie)**	**Flash Player en AIR (laagste voor afspelen vereiste versie)**
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3. Alleen Flash Player 8 en hoger ondersteunen publiceren en afspelen van On2 VP6-video.
H.264 (MPEG-4 Part 10)	9	Flash Player 9 Update 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 Update 3, AIR 1.0
Speex (audio)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

De videobestandsindelingen Adobe F4V en FLV

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Adobe biedt de videobestandsindelingen F4V en FLV voor het streamen van inhoud naar Flash Player en AIR. Zie www.adobe.com/go/video_file_format_nl voor een volledige beschrijving van de videobestandsindelingen.

De videobestandsindeling F4V

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Vanaf Flash Player Update 3 (9.0.115.0) en AIR 1.0 ondersteunen Flash Player en AIR de videobestandsindeling F4V van Adobe die is gebaseerd op de indeling ISO MP4. Subsets van de indeling ondersteunen verschillende functies. Flash Player verwacht dat een geldig F4V-bestand begint met een van de volgende boxes van het hoogste niveau:

- ftyp

  De box ftyp legt vast welke functies een programma moet ondersteunen om een bepaalde bestandsindeling af te spelen.

- moov

  De box moov is in feite de header van een F4V-bestand. Deze box bevat een of meer boxes die op hun beurt andere boxes bevatten die de structuur van de F4V-gegevens vastleggen. Een F4V-bestand moet precies één box moov bevatten.

- mdat

  Een box mdat bevat de gegevens voor het F4V-bestand. Een FV-bestand bevat maar één box mdat. Er moet ook een box moov in het bestand staan omdat de box mdat zonder deze box niet kan worden geïnterpreteerd.

F4V-bestanden ondersteunen gehele getallen van meerdere bytes in de bytevolgorde big endian, waarbij de belangrijkste byte het eerst komt, op het laagste adres.

De videobestandsindeling FLV

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De FLV-bestandsindeling van Adobe bevat gecodeerde audio- en videogegevens die kunnen worden overgedragen door Flash Player. U kunt een encoder, zoals Adobe Media Encoder of Sorenson™ Squeeze, gebruiken om een videobestand van QuickTime of Windows om te zetten naar een FLV-bestand.

Opmerking: *U kunt FLV-bestanden maken door video te importeren in Flash en te exporteren als FLV-bestand. U kunt de FLV Export-insteekmodule gebruiken om FLV-bestanden te exporteren uit ondersteunde videobewerkingstoepassingen. Registreer de bestandsextensie en het MIME-type om FLV-bestanden te kunnen laden vanaf uw webserver. Controleer de documentatie bij de webserver. Het MIME-type voor FLV-bestanden is* `video/x-flv`*. Zie "Informatie over het configureren van FLV-bestanden voor hosting op een server" op pagina 532 voor meer informatie.*

Zie "Geavanceerde onderwerpen voor videobestanden" op pagina 531 voor meer informatie over FLV-bestanden.

Verschillen tussen externe en ingesloten video

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U hebt meer mogelijkheden bij het gebruik van externe videobestanden dan bij het gebruik van geïmporteerde video:

- U kunt langere videoclips gebruiken in uw toepassing zonder dat het afspelen wordt vertraagd. Externe videobestanden maken gebruik van cachegeheugen, dat wil zeggen dat grote bestanden worden opgeslagen in kleine delen die dynamisch kunnen worden geopend. Daarom is voor externe F4V- en FLV-bestanden minder geheugen nodig dan voor ingesloten videobestanden.
- Een extern videobestand kan een andere framesnelheid hebben dan het SWF-bestand waarin de video wordt afgespeeld. U kunt bijvoorbeeld de framesnelheid van het SWF-bestand instellen op 30 frames per seconde (fps) en de framesnelheid van video op 21 fps. Met deze instelling hebt u een betere controle over de video dan bij ingesloten video, zodat u de video vloeiender kunt afspelen. U kunt hiermee tevens videobestanden afspelen bij verschillende framesnelheden zonder de inhoud van het SWF-bestand te wijzigen.
- Bij externe videobestanden hoeft u het afspelen van de SWF-inhoud niet te onderbreken wanneer het videobestand wordt geladen. Bij geïmporteerde videobestanden moet het afspelen van een document soms worden onderbroken om bepaalde functies uit te voeren, zoals het openen van een cd-romstation. Bij videobestanden kunnen functies worden uitgevoerd onafhankelijk van de SWF-inhoud, zonder het afspelen te onderbreken.
- Bij FLV-bestanden kunt u video-inhoud makkelijker van een bijschrift voorzien, omdat u gebeurtenishandlers gebruikt om toegang te krijgen tot de videometagegevens.

De klasse Video

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse Video kunt u livestreamingvideo in een toepassing weergeven zonder deze in het SWF-bestand in te sluiten. U kunt livevideo vastleggen en afspelen met de methode `Camera.getCamera()`. U kunt de klasse Video ook gebruiken om videobestanden af te spelen via HTTP of vanaf het lokale bestandssysteem. U kunt op verschillende manieren video in uw projecten gebruiken:

- Laad een videobestand dynamisch met de klassen NetConnection en NetStream en geef de video weer in een Video-object.

- Leg de invoer van de camera van de gebruiker vast. Zie "Werken met camera's" op pagina 548 voor meer informatie.
- Gebruik de component FLVPlayback.
- Gebruik het besturingselement VideoDisplay.

Opmerking: *Instanties van een Video-object in het werkgebied zijn instanties van de klasse Video.*

Hoewel de klasse Video deel uitmaakt van het pakket flash.media, neemt de klasse Video de gegevens van de klasse flash.display.DisplayObject over. Daarom is alle functionaliteit voor weergaveobjecten, zoals matrixtransformaties en filters, ook van toepassing op Video-instanties.

Zie "Weergaveobjecten manipuleren" op pagina 184, "Werken met geometrie" op pagina 223 en "Weergaveobjecten filteren" op pagina 283 voor meer informatie.

Videobestanden laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het laden van video's met de klassen NetStream en NetConnection is een proces dat uit meerdere stappen bestaat. Als aanbevolen procedure raden wij aan de stappen voor het toevoegen van het Video-object aan de weergavelijst, het bijvoegen van het NetStream-object aan de Video-instantie en het aanroepen van de methode `play()` van het NetStream-object in de opgegeven volgorde uit te voeren:

1 Maak een NetConnection-object. Als u verbinding maakt met een lokaal videobestand of een bestand dat geen gebruik maakt van een server zoals Flash Media Server 2 van Adobe, moet u `null` doorgeven aan de methode `connect()` om de videobestanden af te spelen vanaf een HTTP-adres of een lokaal station. Wanneer u verbinding maakt met een server, stelt u de parameter in op de URI van de toepassing die het videobestand op de server bevat.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
```

2 Maak een nieuw Video-object dat de video weergeeft en voegt het toe aan de weergavelijst van het werkgebied, zoals weergegeven in het volgende fragment:

```
var vid:Video = new Video();
addChild(vid);
```

3 Maak een NetStream-object waarbij het NetConnection-object wordt doorgegeven als een argument naar de constructor. Het volgende fragment verbindt een NetStream-object met de NetConnection-instantie en stelt de gebeurtenishandlers voor de stream in:

```
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 Voeg het NetStream-object toe aan het Video-object met behulp van de methode `attachNetStream()` van het Video-object, zoals weergegeven in het volgende fragment:

```
vid.attachNetStream(ns);
```

5 Roep de methode `play()` van het NetStream-object aan met de videobestands-URL als een argument om het afspelen van de video te starten. Het volgende fragment laadt een videobestand met de naam "video.mp4" en speelt dit af in dezelfde map als het SWF-bestand:

```
ns.play("video.mp4");
```

Meer Help-onderwerpen

Flex: Spark VideoPlayer-besturing

spark.components.VideoDisplay

Video afspelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse NetStream biedt vier hoofdmethoden waarmee u het afspelen van video kunt besturen:

`pause()`: onderbreekt het afspelen van een videostream. Wanneer de video al is gepauzeerd, heeft het aanroepen van deze methode geen effect.

`resume()`: hervat het afspelen van een gepauzeerde videostream. Wanneer de video al wordt afgespeeld, heeft het aanroepen van deze methode geen effect.

`seek()`: hiermee wordt het hoofdframe gezocht dat zich het dichtst bij de opgegeven locatie bevindt (een verschuiving in seconden vanaf het begin van de stream).

`togglePause()`: onderbreekt of hervat het afspelen van een videostream.

Opmerking: *Er is geen methode `stop()`. Wanneer u een stream wilt stoppen, moet u het afspelen pauzeren en zoeken tot het begin van de videostream.*

Opmerking: *Met de methode `play()` wordt het afspelen niet hervat. Deze methode wordt gebruikt om videobestanden te laden.*

In het volgende voorbeeld ziet u hoe u een video kunt besturen met verschillende knoppen. Maak een nieuw document en voeg vier knopinstanties (`pauseBtn`, `playBtn`, `stopBtn` en `togglePauseBtn`) aan het werkgebied toe om het volgende voorbeeld uit te voeren:

```
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();
}
```

Wanneer u tijdens het afspelen van de video op de knopinstantie `pauseBtn` klikt, wordt het videobestand gepauzeerd. Wanneer de video al is gepauzeerd, heeft het klikken op deze knop geen effect. Wanneer u op de knopinstantie `playBtn` klikt, wordt het afspelen hervat als het afspelen was gepauzeerd; anders heeft het klikken op deze knop geen effect.

Einde van een videostream detecteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u naar het begin en einde van een videostream wilt luisteren, moet u een gebeurtenislistener aan de NetStream-instantie toevoegen om naar de gebeurtenis `netStatus` te luisteren. In de volgende code ziet u hoe u naar verschillende codes kunt luisteren tijdens het afspelen van video:

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

De bovenstaande code genereert de volgende uitvoer:

```
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
```

De twee codes waarnaar u specifiek wilt luisteren, zijn 'NetStream.Play.Start' en 'NetStream.Play.Stop', die het begin en het einde van het afspelen van de video aangeven. In het volgende fragment wordt de instructie switch gebruikt om deze twee codes te filteren en een bericht weer te geven:

```
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;
    }
}
```

Wanneer u naar de gebeurtenis `netStatus` (`NetStatusEvent.NET_STATUS`) luistert, kunt u een videospeler maken die de volgende video in een afspeellijst laadt zodra de huidige video klaar is met afspelen.

Video afspelen in de modus Volledig scherm

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met Flash Player en AIR kunt u een toepassing in volledig scherm maken voor het afspelen van video's. Beide toepassingen bieden ook ondersteuning voor het schalen van video's naar het volledige scherm.

Voor AIR-inhoud die in de modus Volledig scherm op het bureaublad wordt uitgevoerd, zijn de schermbeveiligings- en energiebesparingsopties uitgeschakeld tijdens het afspelen van video-inhoud. Dit blijft zo totdat de video stopt of de modus Volledig scherm wordt gesloten.

Zie"Werken met de modus Volledig scherm" op pagina 177 voor alle informatie over het gebruik van de modus Volledig scherm.

Modus Volledig scherm inschakelen voor Flash Player in een browser

Voordat u de modus Volledig scherm kunt implementeren voor Flash Player in een browser, moet u deze voor uw toepassing inschakelen via de sjabloon Publish. Sjablonen die een volledig scherm mogelijk maken, bevatten `<object>`- en `<embed>`-tags die zijn voorzien van de parameter `allowFullScreen`. In het volgende voorbeeld ziet u de parameter `allowFullScreen` in de tag `<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>
```

In Flash selecteert u eerst Bestand -> Publicatie-instellingen en vervolgens de sjabloon Alleen Flash - volledig scherm toestaan op het tabblad HTML van het dialoogvenster Publicatie-instellingen.

In Flex zorgt u dat de HTML-sjabloon is voorzien van `<object>`- en `<embed>`-tags die het volledige scherm ondersteunen.

De modus Volledig scherm starten

Voor Flash Player-inhoud die wordt uitgevoerd in een browser kunt u de modus Volledig scherm voor video starten als reactie op een muisklik of toetsdruk. U kunt bijvoorbeeld instellen dat de modus Volledig scherm wordt gestart wanneer de gebruiker klikt op de knop Volledig scherm of de opdracht Volledig scherm kiest in een contextmenu. Voeg een gebeurtenislistener toe aan het object waarop de actie wordt uitgevoerd, om te reageren op de gebruiker. De volgende code voegt een gebeurtenislistener toe aan een knop waarop de gebruiker klikt om de modus Volledig scherm te activeren:

```
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;

}
```

De code start de modus Volledig scherm door de eigenschap `Stage.displayState` in te stellen op `StageDisplayState.FULL_SCREEN`. Deze code schaalt het volledige werkgebied naar het volledige scherm waarbij de video wordt geschaald in verhouding met de ruimte die deze inneemt in het werkgebied.

Met de eigenschap `fullScreenSourceRect` kunt u opgeven dat een bepaald gebied van het werkgebied moet worden geschaald naar het volledige scherm. Definieer eerst de rechthoek die u wilt schalen naar het volledige scherm. Wijs de rechthoek vervolgens toe aan de eigenschap `Stage.fullScreenSourceRect`. Deze versie van de functie `fullScreenButtonHandler()` voegt twee extra coderegels toe die alleen de video schalen naar het volledige scherm.

```
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;
}
```

Hoewel in dit voorbeeld een gebeurtenishandler wordt aangeroepen als reactie op een muisklik, is de techniek voor het activeren van de modus Volledig scherm hetzelfde voor Flash Player en AIR. Definieer de rechthoek die u wilt schalen en stel vervolgens de eigenschap `Stage.displayState` in. Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

In het volledige voorbeeld, dat hieronder volgt, is code toegevoegd die de verbinding en het NetStream-object voor de video maakt, en die de video begint af te spelen.

```
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
        }
    }
}
```

De functie `onMetaData()` is een callbackfunctie voor het afhandelen van videometagegevens, als deze aanwezig zijn. Een callbackfunctie is een functie die door de runtime wordt aangeroepen als reactie op een bepaald type gebeurtenis of optreden. In dit voorbeeld is de functie `onMetaData()` een stub die voldoet aan de vereisten om de functie te leveren. Zie "Callbackmethoden schrijven voor metagegevens en actiepunten" op pagina 512 voor meer informatie.

Modus Volledig scherm beëindigen

Een gebruiker kan de modus Volledig scherm beëindigen met een van de sneltoetsen, zoals de Escape-toets. U beëindigt de modus Volledig scherm in ActionScript door de eigenschap `Stage.displayState` in te stellen op `StageDisplayState.NORMAL`. De code in het volgende voorbeeld beëindigt de modus Volledig scherm wanneer de NetStream.Play.Stop-gebeurtenis `netStatus` optreedt.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

Hardwareversnelling voor volledig scherm

Wanneer een rechthoekig segment van het werkgebied wordt geschaald naar het volledige scherm, gebruikt Flash Player of AIR hardwareversnelling, op voorwaarde dat deze beschikbaar en ingeschakeld is. De runtime gebruikt de grafische kaart van de computer om het schalen van de video, of een deel van het werkgebied, naar het volledige scherm te versnellen. In deze omstandigheden kunnen Flash Player-toepassingen vaak hun voordeel halen uit het schakelen naar de StageVideo-klasse vanuit de Video-klasse (of Camera-klasse; Flash Player 11.4/AIR 3.4 en hoger).

Zie "Werken met de modus Volledig scherm" op pagina 177 voor informatie over hardwareversnelling in de modus Volledig scherm. Zie "De klasse StageVideo gebruiken voor presentatie met hardwareversnelling" op pagina 539 voor meer informatie over StageVideo.

Videobestanden streamen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u bestanden vanaf Flash Media Server wilt streamen, kunt u de klassen NetConnection en NetStream gebruiken om verbinding te maken met een externe serverinstantie en een opgegeven stream af te spelen. Wanneer u een RTMP-server (Real Time Messaging Protocol) wilt opgeven, geeft u de gewenste RTMP-URL, zoals 'rtmp://localhost/appName/appInstance', aan de methode `NetConnection.connect()` door, in plaats van dat u de waarde `null` doorgeeft. Wanneer u een bepaalde livestream of opgenomen stream vanaf de opgegeven Flash Media Server wilt afspelen, geeft u een identificerende naam voor livegegevens die door de methode `NetStream.publish()` worden gepubliceerd, of een bestandsnaam van opgenomen gegevens die kunnen worden afgespeeld, door aan de methode `NetStream.play()`.

Video naar een server verzenden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u complexere toepassingen met Video- en Camera-objecten wilt maken, biedt Flash Media Server een combinatie van streamingmedia en een ontwikkelomgeving voor het maken en leveren van mediatoepassingen aan een breed publiek. Met deze combinatie kunnen ontwikkelaars toepassingen maken zoals video op aanvraag, livewebuitzendingen en MP3-streams of videoblogs, videomessaging en multimediachatomgevingen. Zie de onlinedocumentatie bij Flash Media Server op www.adobe.com/go/learn_fms_docs_nl voor meer informatie.

Actiepunten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt actiepunten tijdens het coderen insluiten in een Adobe F4V- of FLV-videobestand. In het verleden werden actiepunten in films ingesloten om de filmoperateur een visueel signaal te geven dat de filmrol bijna op was. In de video-indelingen Adobe F4V en FLV kunt u actiepunten gebruiken om een of meer acties in de toepassing te starten op het moment dat een actiepunt voorkomt in de videostream.

U kunt meerdere verschillende soorten actiepunten gebruiken in Flash-video. U kunt ActionScript gebruiken om te interageren met actiepunten die u in een videobestand insluit wanneer u dit maakt.

- Navigatieactiepunten: u kunt navigatieactiepunten in het pakket met videostream en metagegevens insluiten wanneer u het videobestand codeert. U kunt navigatieactiepunten gebruiken om gebruikers te laten zoeken naar een bepaald deel van een bestand.
- Gebeurtenisactiepunten: u kunt gebeurtenisactiepunten in het pakket met videostream en metagegevens insluiten wanneer u het videobestand codeert. U kunt code schrijven om de gebeurtenissen af te handelen die op bepaalde punten worden geactiveerd tijdens het afspelen van het videobestand.

- ActionScript-actiepunten: ActionScript-actiepunten zijn alleen beschikbaar voor de Flash-component FLVPlayback. ActionScript-actiepunten zijn externe actiepunten die u met ActionScript-code kunt maken en benaderen. U kunt code schrijven om deze actiepunten te activeren met betrekking tot het afspelen van video. Deze actiepunten zijn minder nauwkeurig dan ingesloten actiepunten (tot een tiende van een seconde), omdat videospelers deze afzonderlijk bijhouden. Wanneer u een toepassing wilt maken waarin u gebruikers naar een actiepunt wilt laten navigeren, moet u actiepunten maken en insluiten wanneer u het bestand codeert in plaats van gebruik te maken van ActionScript-actiepunten. U wordt aangeraden de actiepunten in het FLV-bestand in te sluiten omdat ze nauwkeuriger zijn.

Bij navigatieactiepunten wordt een hoofdframe gemaakt op de opgegeven actiepuntlocatie, zodat u met code de afspeelkop van een videospeler naar deze locatie kunt verplaatsen. U kunt bepaalde punten in een videobestand instellen op de plaatsen waar u wilt dat gebruikers gaan zoeken. U hebt bijvoorbeeld een video met meerdere hoofdstukken of segmenten en u kunt de video besturen door navigatieactiepunten in te sluiten in het videobestand.

Zie "Actiepunten insluiten" in *Flash gebruiken* voor meer informatie over het coderen van Adobe-videobestanden met actiepunten.

U kunt actiepuntparameters openen door ActionScript te schrijven. Actiepuntparameters maken deel uit van het gebeurtenisobject dat is ontvangen door de callbackhandler.

Als u bepaalde handelingen in uw code wilt activeren wanneer een FLV-bestand een opgegeven actiepunt heeft bereikt, gebruikt u de gebeurtenishandler `NetStream.onCuePoint`.

Om een actie voor een actiepunt in een F4V-videobestand te synchroniseren, moet u de actiepuntgegevens ophalen van de callbackfunctie `onMetaData()` of `onXMPData()` en het actiepunt starten met de klasse Timer in ActionScript 3.0. Zie "onXMPData() gebruiken" op pagina 524 voor meer informatie over F4V-actiepunten..

Zie "Callbackmethoden schrijven voor metagegevens en actiepunten" op pagina 512 voor meer informatie over het werken met actiepunten en metagegevens.

Callbackmethoden schrijven voor metagegevens en actiepunten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt handelingen in uw toepassing activeren wanneer bepaalde metagegevens zijn ontvangen of bepaalde actiepunten zijn bereikt door de speler. Wanneer deze gebeurtenissen optreden, moet u bepaalde callbackmethoden gebruiken als gebeurtenishandlers. De klasse NetStream geeft de volgende metagegevensgebeurtenissen op die kunnen optreden tijdens het afspelen: `onCuePoint` (alleen FLV-bestanden), `onImageData`, `onMetaData`, `onPlayStatus`, `onTextData` en `onXMPData`.

U moet callbackmethoden voor deze handlers maken. Anders kunnen er fouten optreden in Flash-runtime. De volgende code speelt bijvoorbeeld het FLV-bestand video.flv in dezelfde map als het SWF-bestand af:

```
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);
```

De vorige code laadt het lokale videobestand video.flv en luistert of `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) wordt verzonden. Deze gebeurtenis wordt verzonden wanneer een uitzonderingsfout wordt gegenereerd vanuit native asynchrone code. In dit geval wordt de gebeurtenis verzonden wanneer het videobestand metagegevens of informatie over actiepunten bevat, en de juiste listeners niet zijn gedefinieerd. De vorige code handelt de gebeurtenis `asyncError` af en negeert de fout wanneer u niet bent geïnteresseerd in de metagegevens of de informatie over actiepunten van het videobestand. Wanneer u een FLV-bestand met metagegevens en verschillende actiepunten had, zou de functie `trace()` de volgende foutberichten weergeven:

```
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.
```

De fouten treden op omdat het NetStream-object de callbackmethode `onMetaData` of `onCuePoint` niet kon vinden. U kunt deze callbackmethoden op verschillende manieren in uw toepassingen definiëren.

Meer Help-onderwerpen

Flash Media Server: metagegevens verwerken in streams

De eigenschap client van het NetStream-object op een object instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de eigenschap `client` op een object of een subklasse van NetStream instelt, kunt u de callbackmethoden `onMetaData` en `onCuePoint` doorsturen of volledig negeren. In het volgende voorbeeld ziet u hoe u een leeg object kunt gebruiken om de callbackmethoden te negeren, zonder naar de gebeurtenis `asyncError` te luisteren:

```
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);
```

Wanneer u naar de callbackmethode `onMetaData` of `onCuePoint` wilt luisteren, moet u methoden definiëren om die callbackmethoden af te handelen. Dit wordt in het volgende fragment getoond:

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

De vorige code luistert naar de callbackmethode `onMetaData` en roept de methode `metaDataHandler()` op, die een tekenreeks traceert. Als de Flash-runtime een actiepunt tegenkomt, treden er geen fouten op, zelfs als geen callbackmethode `onCuePoint` is gedefinieerd.

Aangepaste klassen maken en methoden definiëren voor het afhandelen van callbackmethoden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De volgende code stelt de eigenschap `client` van het NetStream-object in op de aangepaste klasse CustomClient, die handlers voor de callbackmethoden definieert:

```
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);
```

De klasse CustomClient ziet er als volgt uit:

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

De klasse CustomClient definieert een handler voor de callbackhandler `onMetaData`. Wanneer Flash Player een actiepunt tegenkomt en de callbackhandler `onCuePoint` is aangeroepen, wordt de gebeurtenis `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) verzonden om aan te geven dat flash.net.NetStream de callback onCuePoint niet kon aanroepen. U kunt deze fout voorkomen door de callbackmethode `onCuePoint` in de klasse CustomClient te definiëren of door een gebeurtenishandler voor de gebeurtenis `asyncError` te definiëren.

De klasse NetStream uitbreiden en methoden toevoegen voor het afhandelen van callbackmethoden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de volgende code wordt een instantie van de klasse CustomNetStream gemaakt die in een later codevoorbeeld wordt gedefinieerd:

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Het volgende codevoorbeeld definieert de klasse CustomNetStream die de klasse NetStream uitbreidt, het maken van het benodigde NetConnection-object afhandelt en de methoden `onMetaData` en `onCuePoint` voor de callbackhandler afhandelt:

```
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");
        }
    }
}
```

Wanneer u de methoden `onMetaData()` en `onCuePoint()` in de klasse CustomNetStream wilt hernoemen, kunt u de volgende code gebruiken:

```
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");
        }
    }
}
```

De klasse NetStream uitbreiden en dynamisch maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse NetStream uitbreiden en de subklasse dynamisch maken, zodat de callbackhandlers `onCuePoint` en `onMetaData` dynamisch kunnen worden toegevoegd. Dit wordt geïllustreerd in het volgende codevoorbeeld:

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

De klasse DynamicCustomNetStream ziet er als volgt uit:

```
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);
        }
    }
}
```

Er worden zelfs geen fouten gegenereerd wanneer de callbackhandlers `onMetaData` en `onCuePoint` geen handlers bevatten, omdat de klasse DynamicCustomNetStream dynamisch is. Wanneer u de methoden `onMetaData()` en `onCuePoint()` voor de callbackhandlers wilt hernoemen, kunt u de volgende code gebruiken:

```
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");
}
```

De eigenschap client van het NetStream-object op this instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de eigenschap `client` op `this` instelt, zoekt de toepassing in het huidige bereik naar de methoden `onMetaData()` en `onCuePoint()`. Dit wordt in het volgende voorbeeld geïllustreerd:

```
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);
```

Wanneer de callbackhandler `onMetaData` of `onCuePoint` wordt aangeroepen en er geen methoden voor het afhandelen van de callback bestaan, worden er door Flash Player geen fouten gegenereerd. Wanneer u deze callbackhandlers wilt afhandelen, kunt u de methoden `onMetaData()` en `onCuePoint()` in uw code maken. Dit wordt in het volgende codevoorbeeld geïllustreerd:

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

Actiepunten en metagegevens gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de NetStream-callbackmethoden kunt u actiepunt- en metagegevensgebeurtenissen vastleggen en verwerken terwijl de video wordt afgespeeld.

Actiepunten gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de volgende tabel worden de callbackmethoden beschreven die u kunt gebruiken om F4V- en FLV-actiepunten vast te leggen in Flash Player en AIR.

Runtime	F4V	FLV
Flash Player 9 / AIR 1.0		OnCuePoint
		OnMetaData
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

In het volgende voorbeeld wordt een eenvoudige `for...in`-lus gebruikt om een bewerking uit te voeren op elke eigenschap in de parameter `infoObject` die de functie `onCuePoint()` ontvangt. De lus roept de functie trace() op om een bericht weer te geven wanneer actiepuntgegevens worden ontvangen:

```
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]);
    }
}
```

De volgende uitvoer wordt weergegeven:

```
parameters:
name: point1
time: 0.418
type: navigation
```

Deze code maakt gebruik van een van de vele technieken voor het instellen van het object waarop de callbackmethode wordt uitgevoerd. U kunt andere technieken gebruiken, zie "Callbackmethoden schrijven voor metagegevens en actiepunten" op pagina 512 voor meer informatie.

Videometagegevens gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de functies `OnMetaData()` and `OnXMPData()` gebruiken om informatie over metagegevens, inclusief actiepunten, te benaderen in het videobestand.

OnMetaData() gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Metagegevens bevatten informatie over het videobestand, zoals duur, breedte, hoogte en framesnelheid. Welke informatie over metagegevens wordt toegevoegd aan het videobestand, is afhankelijk van de software die u gebruikt om het videobestand te coderen.

```
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]);
    }
}
```

De bovenstaande code genereert de volgende uitvoer:

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

Wanneer u video zonder audio hebt, wordt de informatie over metagegevens betreffende audio (zoals `audiodatarate`*) geretourneerd als* `undefined` *omdat geen audio-informatie aan de metagegevens is toegevoegd tijdens het coderen.*

In de bovenstaande code wordt de actiepuntinformatie niet weergegeven. U kunt de volgende functie gebruiken die de items in een object recursief weergeeft om de metagegevens van de actiepunten weer te geven:

```
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);
        }
    }
}
```

Wanneer u het vorige codefragment gebruikt om de parameter `infoObject` in de methode `onMetaData()` weer te geven, wordt de volgende uitvoer gegenereerd:

```
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
```

Het volgende voorbeeld geeft de metagegevens voor een MP4-video weer. Hierbij wordt ervan uitgegaan dat er een TextArea-object is met de naam `metaDataOut`, waarnaar de metagegevens worden geschreven.

```
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;
        }
    }
}
```

De functie `onMetaData()` heeft de volgende uitvoer geproduceerd voor deze video:

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

Het informatieobject gebruiken

In de volgende tabel worden de mogelijke waarden getoond voor videometagegevens die worden doorgegeven aan de callbackfunctie `onMetaData()` in het object dat ze ontvangen:

Parameter	Beschrijving
aacaot	Objecttype AAC-audio; 0, 1 en 2 worden ondersteund.
avclevel	AVC IDC-niveaunummer zoals 10, 11, 20, 21 enzovoort.
avcprofile	AVC-profielnummer zoals 55, 77, 100 enzovoort.
audiocodecid	Een tekenreeks die de gebruikte audio-codec aangeeft (techniek voor coderen/decoderen), bijvoorbeeld .mp3 of mp4a.
audiodatarate	Een getal dat de snelheid aangeeft waarmee audio wordt gecodeerd, in kilobytes per seconde.
audiodelay	Een getal dat aangeeft op welk tijdstip in het FLV-bestand 'tijd 0' van het oorspronkelijke FLV-bestand valt. De video-inhoud moet met een kleine hoeveelheid worden vertraagd voor correcte synchronisatie van de audio.
canSeekToEnd	Een Booleaanse waarde die `true` is als het FLV-bestand met een hoofdframe op het laatste frame is gecodeerd, waardoor u naar het einde van een filmfragment met progressieve download kunt zoeken. Wanneer het FLV-bestand niet met een hoofdframe op het laatste frame is gecodeerd, is de waarde `false`.
cuePoints	Een array van objecten, één voor elk actiepunt dat in het FLV-bestand is ingesloten. Als het FLV-bestand geen actiepunten bevat, is de waarde ongedefinieerd. Elk object heeft de volgende eigenschappen: • `type`: een tekenreeks die het type actiepunt aangeeft, zoals 'navigatie' of 'gebeurtenis'. • `name`: een tekenreeks die de naam van het actiepunt aangeeft. • `time`: een getal dat de tijd van het actiepunt aangeeft, opgegeven in seconden en met een precisie van drie decimalen (milliseconden). • `parameters`: een optioneel object dat naam/waarde-paren bevat die door de gebruiker worden opgegeven bij het maken van actiepunten.
duration	Een getal dat de duur van het videobestand opgeeft, in seconden.
framerate	Een getal dat de framesnelheid van het FLV-bestand aangeeft.
height	Een getal dat de hoogte van het FLV-bestand aangeeft, in pixels.
seekpoints	Een array van de beschikbare hoofdframes als tijdstempels in milliseconden. Optioneel.
tags	Een array van sleutelwaardeparen die de informatie in het atoom "ilst" vertegenwoordigt. Dit is het equivalent van ID3-tags voor MP4-bestanden. iTunes maakt gebruikt van deze tags. Kan worden gebruikt voor de weergave van illustraties, als deze beschikbaar zijn.
trackinfo	Een object dat informatie levert over alle tracks in het MP4-bestand, inclusief de voorbeeldbeschrijvings-id.
videocodecid	Een tekenreeks die de codec-versie aangeeft die is gebruikt om de video te coderen. - bijvoorbeeld "avc1" of "VP6F"

Parameter	Beschrijving
videodatarate	Een getal dat de videogegevenssnelheid van het FLV-bestand aangeeft.
videoframerate	De framesnelheid van de MP4-video.
width	Een getal dat de breedte van het FLV-bestand aangeeft, in pixels.

In de volgende tabel worden de mogelijke waarden voor de parameter `videocodecid` weergegeven:

videocodecid	Codec-naam
2	Sorenson H.263
3	Schermvideo (alleen SWF versie 7 en hoger)
4	VP6 (alleen SWF versie 8 en hoger)
5	VP6-video met alfakanaal (alleen SWF versie 8 en hoger)

In de volgende tabel worden de mogelijke waarden voor de parameter `audiocodecid` weergegeven:

audiocodecid	Codec-naam
0	geen compressie
1	ADPCM
2	MP3
4	Nellymoser @ 16 kHz mono
5	Nellymoser, 8 kHz mono
6	Nellymoser
10	AAC
11	Speex

onXMPData() gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

De callbackfunctie `onXMPData()` ontvangt specifieke Adobe XMP-gegevens (Extensible Metadata Platform) die zijn ingesloten in het Adobe F4V- of FLV-videobestand. De XMP-metagegevens bevatten zowel actiepunten als andere videometagegevens. Ondersteuning voor XMP-metagegevens wordt geïntroduceerd met Flash Player 10 en Adobe AIR 1.5 en wordt door de daaropvolgende versies ondersteund.

In het volgende voorbeeld worden actiepuntgegevens in de XMP-metagegevens verwerkt:

```
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);
        }
    }
}
```

Voor een kort videobestand met de naam startrekintro.f4v maakt dit voorbeeld de volgende traceerregels. De regels geven de actiepuntgegevens voor navigatie en de gebeurtenisactiepunten in de XMP-metagegevens aan:

```
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
```

Opmerking: *In XMP-gegevens wordt tijd opgeslagen als DVA Ticks, en niet als seconden. Voor de berekening van de actiepunttijd deelt u de begintijd door de framesnelheid. De begintijd 7695905817600 gedeeld door een framesnelheid van 254016000000 is gelijk aan 30:30.*

Als u de volledige, onbewerkte XMP-metagegevens wilt bekijken, die ook de framesnelheid bevatten, verwijdert u de aanduidingen voor opmerkingen (//) die vóór de tweede en derde `trace()`-instructie aan het begin van de functie `onXMPData()` staan.

Voor meer informatie over XMP zie:

- http://partners.adobe.com/public/developer/xmp/topic.html
- http://www.adobe.com/devnet/xmp/

Metagegevens van afbeeldingen gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De gebeurtenis `onImageData` verzendt afbeeldingsgegevens als een bytearray via een AMF0-gegevenskanaal. De gegevens kunnen in de indeling JPEG, PNG of GIF zijn. Definieer de callbackmethode `onImageData()` voor de verwerking van deze gegevens op dezelfde manier als u callbackmethoden zou definiëren voor `onCuePoint` en `onMetaData`. In het volgende voorbeeld wordt de callbackmethode `onImageData()` gebruikt om afbeeldingsgegevens te benaderen en weer te geven:

```
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);
}
```

Tekstmetagegevens gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De gebeurtenis `onTextData` verzendt tekstgegevens via een AMF0-gegevenskanaal. De tekstgegevens hebben de UTF-8-indeling en bevatten aanvullende informatie over opmaak op basis van de 3GP-specificatie voor getimede tekst. Deze specificatie legt een gestandaardiseerde indeling voor ondertitels vast. Definieer de callbackmethode `onTextData()` voor de verwerking van deze gegevens op dezelfde manier als u callbackmethoden zou definiëren voor `onCuePoint` of `onMetaData`. In het volgende voorbeeld geeft de methode `onTextData()` de track-id en de bijbehorende tracktekst weer.

```
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);
}
```

Toezicht houden op NetStream-activiteit

Flash Player 10.3 of hoger, Adobe AIR 2.7 of hoger

U kunt toezicht houden op NetStream-activiteit en op die manier de gegevens verzamelen die vereist zijn voor de analyse en rapportage van mediagebruik. Met de toezichtfuncties die in dit gedeelte aan de orde komen, kunt u mediametingbibliotheken maken waarmee gegevens worden verzameld zonder close-coupling naar de specifieke videospeler waarop de media worden weergegegeven. Op die manier kunnen uw clientontwikkelaars hun favoriete videospelers kiezen wanneer ze uw bibliotheek gebruiken. Gebruik de klasse NetMonitor om toezicht te houden op het maken en de activiteit van NetStream-objecten in een toepassing. De klasse NetMonitor biedt een lijst met de actieve NetStreams die op een bepaald moment bestaan en verzendt eveneens een gebeurtenis telkens wanneer een NetStream-object wordt gemaakt.

Een NetStream-object verzendt de gebeurtenissen die zijn opgenomen in de volgende tabel, op basis van het mediatype dat wordt afgespeeld:

Gebeurtenis	Progressieve download	RTMP-streaming	HTTP-streaming
NetStream.Play.Start	Ja	Ja	Nee
NetStream.Play.Stop	Ja	Ja	Nee
NetStream.Play.Complete	Ja	Ja	Nee
NetStream.SeekStart.Notify	Ja	Ja	Ja
NetStream.Seek.Notify	Ja	Ja	Ja
NetStream.Unpause.Notify	Ja	Ja	Ja
NetStream.Unpause.Notify	Ja	Ja	Ja
NetStream.Play.Transition	Niet van toepassing	Ja	Niet van toepassing
NetStream.Play.TransitionComplete	Niet van toepassing	Ja	Niet van toepassing
NetStream.Buffer.Full	Ja	Ja	Ja
NetStream.Buffer.Flush	Ja	Ja	Ja
NetStream.Buffer.Empty	Ja	Ja	Ja

Het NetStreamInfo-object dat is gekoppeld aan een NetStream-instantie slaat eveneens de laatste metagegevens en XMP-gegevensobjecten op die in de media zijn aangetroffen.

Als media worden afgespeeld via HTTP-streaming, worden de NetStream.Play.Start, NetStream.Play.Stop en NetStream.Play.Complete niet verzonden omdat de toepassing volledige controle heeft over de mediastream. In dat geval moet een videospeler deze gebeurtenissen voor HTTP-streams synthetiseren en verzenden.

NetStream.Play.Transition en NetStream.Play.TransitionComplete worden evenmin verzonden voor progressieve download of HTTP-media. Dynamisch schakelen tussen bitsnelheden is een RTMP-functie. Als een videospeler die gebruikmaakt van een HTTP-stream ondersteuning biedt voor een vergelijkbare functie, kan de speler overgangsgebeurtenissen synthetiseren en verzenden.

Meer Help-onderwerpen

Adobe Developer Connection: Measuring video consumption in Flash

Toezicht houden op NetStream-gebeurtenissen

Twee typen gebeurtenissen bieden waardevolle gebruiksgegevens: `netStatus` en `mediaTypeData`. Daarnaast kan een timer worden gebruikt om van tijd tot tijd de positie van de NetStream-afspeelkop vast te leggen.

`netStatus`-gebeurtenissen bieden informatie die u kunt gebruiken om te bepalen hoeveel van een stream er al door een gebruiker is weergegeven. Buffer- en RTMFP-streamovergangsgebeurtenissen resulteren eveneens in een `netStatus`-gebeurtenis.

`mediaTypeData`-gebeurtenissen bieden informatie over meta- en XMP-gegevens. De Netstream.Play.Complete-gebeurtenis wordt verzonden als een `mediaTypeData`-gebeurtenis. Andere gegevens die in de stream zijn ingesloten, zijn ook beschikbaar via `mediaTypeData`-gebeurtenissen, met inbegrip van cuepunten, tekst en afbeeldingen.

In het volgende voorbeeld wordt getoond hoe een klasse kan worden gemaakt die toezicht houdt op de status en gegevensgebeurtenissen vanuit alle actieve NetStreams in een toepassing. Een dergelijke klasse zou doorgaans de relevante gegevens uploaden voor analyse naar een server voor verzamelingsdoeleinden.

```
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
            }
        }

    }
}
```

Spelerdomein vaststellen

De URL en het domein van de webpagina waarop een gebruiker mediacontent weergeeft, zijn niet altijd gemakkelijk te achterhalen. Indien dit is toegestaan door de hostwebsite, kunt u de klasse ExternalInterface gebruiken om de exacte URL te verkrijgen. Het is echter mogelijk dat bepaalde websites het gebruik van de ExternalInterface door videospelers van derden niet toestaan. In dergelijke gevallen kunt u het domein van de actieve wegbpagina achterhalen via de `pageDomain`-eigenschap van de klasse Security. De volledige URL wordt uit gebruikersbeveiligings- en privacyoverwegingen niet vrijgegeven.

Het paginadomein kan worden verkregen via de statische `pageDomain`-eigenschap van de klasse Security:

```
var domain:String = Security.pageDomain;
```

Geavanceerde onderwerpen voor videobestanden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De volgende onderwerpen gaan over een aantal specifieke problemen bij het werken met FLV-bestanden.

Informatie over het configureren van FLV-bestanden voor hosting op een server

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u met FLV-bestanden werkt, moet u de server configureren om te kunnen werken met de bestandsindeling FLV. MIME (Multipurpose Internet Mail Extensions) is een gestandaardiseerde gegevensspecificatie waarmee u niet-ASCII-bestanden via internetverbindingen kunt verzenden. Webbrowsers en e-mailclients worden geconfigureerd om diverse MIME-typen te interpreteren, zodat deze video, audio, afbeeldingen en opgemaakte tekst kunnen verzenden en ontvangen. Mogelijk moet u de bestandsextensie en het MIME-type registreren om FLV-bestanden te kunnen laden vanaf uw webserver. Raadpleeg daarom de documentatie bij de webserver. Het MIME-type voor FLV-bestanden is `video/x-flv`. De volledige informatie over het bestandstype FLV is als volgt:

- Mime-type: video/x-flv
- Bestandsextensie: .flv
- Vereiste parameters: geen
- Optionele parameters: geen
- Coderingsoverwegingen: FLV-bestanden zijn binaire bestanden. In sommige toepassingen moet u het subtype application/octet-stream instellen.
- Beveiligingsproblemen: geen
- Gepubliceerde specificatie: www.adobe.com/go/video_file_format_nl

De wijze waarop streamingmedia door Microsoft worden afgehandeld via de webserver van Microsoft IIS (Internet Information Services) 6.0 is anders dan in lagere versies. Bij eerdere versies van IIS is geen enkele wijziging nodig voor het streamen van Flash Video. In IIS 6.0, de standaardwebserver van Windows 2003, is een MIME-type voor de server vereist om FLV-bestanden te kunnen herkennen als streamingmedia.

Wanneer SWF-bestanden die externe FLV-bestanden streamen, op een server met Microsoft Windows Server® 2003 worden geplaatst en in een browser worden weergegeven, wordt het SWF-bestand correct afgespeeld, maar streamt de FLV-video niet. Dit probleem heeft gevolgen voor alle FLV-bestanden die op de server met Windows Server 2003 zijn geplaatst, inclusief de bestanden die u maakt met lagere versies van het Flash-programma voor het schrijven van programmacode en met Macromedia Flash Video Kit voor Dreamweaver MX 2004 van Adobe. Deze bestanden werken correct wanneer u deze test op andere besturingssystemen.

Zie www.adobe.com/go/tn_19439_nl voor meer informatie over het configureren van Microsoft Windows 2003 en Microsoft IIS Server 6.0 voor het streamen van FLV-video.

Informatie over het plaatsen van lokale FLV-bestanden op de Macintosh

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een lokaal FLV-bestand vanaf een niet-systeemstation op een Apple® Macintosh® probeert af te spelen via een pad dat een relatieve slash (/) bevat, wordt de video niet afgespeeld. Onder niet-systeemstations vallen onder andere cd-romstations, gepartitioneerde vaste schijven, verwisselbare opslagmedia en aangesloten opslagapparaten.

Opmerking: *Dit probleem is het gevolg van een beperking in het besturingssysteem en niet van een beperking in Flash Player of AIR.*

Als u een FLV-bestand wilt afspelen vanaf een niet-systeemstation op een Mac, kunt u het best hiernaar verwijzen met een absoluut pad waarbij u gebruikmaakt van een dubbelepuntnotatie (:) in plaats van slash-notatie (/). In de volgende lijst kunt u het verschil zien tussen de twee soorten notaties:

- Slash-notatie**:** mijnstation/mijnmap/mijnFLV.flv
- Dubbelepuntnotatie**:** (Mac OS®) mijnstation:mijnmap:mijnFLV.flv

Voorbeeld van video: videojukebox

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het volgende voorbeeld wordt een eenvoudige videojukebox gemaakt waarin een lijst van video's dynamisch wordt geladen en op volgorde wordt afgespeeld. Hierdoor kunt u een toepassing maken waarmee een gebruiker door een reeks videozelfstudies kan bladeren. Er kan ook worden opgegeven welke advertenties moeten worden afgespeeld voordat de video wordt afgespeeld die de gebruiker heeft aangevraagd. Dit voorbeeld demonstreert de volgende functies van ActionScript 3.0:

- Afspeelkoppen bijwerken op basis van de voortgang van het afspelen van een videobestand
- Luisteren naar en parseren van metagegevens van een videobestand
- Opgegeven codes afhandelen in een netwerkstream
- Dynamisch geladen FLV-bestanden laden, afspelen, pauzeren en stoppen
- De grootte van een Video-object wijzigen in de weergavelijst, gebaseerd op de metagegevens van de netwerkstream

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van de videojukebox vindt u in de map Samples/Videojukebox. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
VideoJukebox.fla of VideoJukebox.mxml	Het hoofdtoepassingsbestand voor Flex (MXML) of Flash (FLA).
VideoJukebox.as	De klasse die de hoofdfunctionaliteit van de toepassing biedt.
playlist.xml	Een bestand dat weergeeft welke videobestanden in de videojukebox worden geladen.

Externe bestanden voor de videoafspeellijst laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het externe bestand playlist.xml geeft aan welke video's moeten worden geladen en in welke volgorde ze moeten worden afgespeeld. Wanneer u het XML-bestand wilt laden, moet u een URLLoader- en een URLRequest-object gebruiken, zoals in het volgende voorbeeld wordt geïllustreerd:

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

Deze code wordt in de constructor van de klasse VideoJukebox geplaatst, zodat het bestand wordt geladen voordat andere codes worden uitgevoerd. Zodra het XML-bestand klaar is met laden, wordt de methode `xmlCompleteHandler()` aangeroepen, die het externe bestand in een XML-object parseert. Dit wordt in het volgende voorbeeld geïllustreerd:

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

Het XML-object `playlist` bevat de onbewerkte XML-gegevens van het externe bestand, terwijl videos XML een XMLList-object is dat alleen de videonodes bevat. In het volgende fragment wordt het voorbeeldbestand playlist.xml weergegeven:

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

Ten slotte roept de methode `xmlCompleteHandler()` de methode `main()` op, waarmee de verschillende componentinstanties in de weergavelijst worden ingesteld. Bovendien worden de NetConnection- en NetStream-objecten ingesteld die worden gebruikt om de externe FLV-bestanden te laden.

De gebruikersinterface maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de gebruikersinterface wilt maken, moet u vijf Button-instanties naar de weergavelijst slepen en ze de volgende instantienamen geven: `playButton`, `pauseButton`, `stopButton`, `backButton` en `forwardButton`.

Voor elk van deze Button-instanties moet u een handler toewijzen voor de gebeurtenis `click`, zoals in het volgende fragment wordt geïllustreerd:

```
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);
```

De methode `buttonClickHandler()` gebruikt de instructie switch om te bepalen op welke Button-instantie is geklikt, zoals in de volgende code wordt geïllustreerd:

```
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;
    }
}
```

Vervolgens voegt u een Slider-instantie aan de weergavelijst toe en geeft u deze de instantienaam `volumeSlider`. De volgende code stelt de eigenschap `liveDragging` van de Slider-instantie in op `true` en definieert een gebeurtenislistener voor de gebeurtenis `change` van de Slider-instantie:

```
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);
```

Voeg een ProgressBar-instantie aan de weergavelijst toe en geef deze de instantienaam `positionBar`. Stel de eigenschap `mode` ervan in op manual, zoals in het volgende fragment wordt geïllustreerd:

```
positionBar.mode = ProgressBarMode.MANUAL;
```

Voeg ten slotte een Label-instantie aan de weergavelijst toe en geef deze de instantienaam `positionLabel`. De waarde van deze Label-instantie wordt door de Timer-instantie ingesteld.

Luisteren naar metagegevens van een Video-object

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer Flash Player metagegevens van de geladen video's tegenkomt, wordt de callbackhandler `onMetaData()` op de eigenschap `client` van het NetStream-object aangeroepen. De volgende code initialiseert een object en stelt de opgegeven callbackhandler in:

```
client = new Object();
client.onMetaData = metadataHandler;
```

De methode `metadataHandler()` kopieert de gegevens naar de eigenschap meta, die eerder in de code is gedefinieerd. Hierdoor hebt u in de hele toepassing toegang tot de metagegevens van de huidige video. Vervolgens worden de afmetingen van het Video-object op het werkgebied gewijzigd, zodat deze met de geretourneerde afmetingen van de metagegevens overeenkomen. Ten slotte wordt de positionBar-voortgangsbalkinstantie verplaatst en worden de afmetingen ervan gewijzigd, gebaseerd op de afmetingen van de video die momenteel wordt afgespeeld. De volgende code bevat de gehele methode `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;
}
```

Een video dynamisch laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing gebruikt een NetConnection- en een NetStream-object om elk van de video's dynamisch te laden. De volgende code maakt een NetConnection-object en geeft `null` door aan de methode `connect()`. Door `null` op te geven, maakt Flash Player verbinding met een video op de lokale server, in plaats van verbinding te maken met een server, zoals Flash Media Server.

De volgende code maakt zowel de NetConnection- als de NetStream-instantie, definieert een gebeurtenislistener voor de gebeurtenis `netStatus` en wijst het `client`-object aan de eigenschap `client` toe:

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

De methode `netStatusHandler()` wordt aangeroepen wanneer de status van de video is gewijzigd. Dat wil zeggen, wanneer een video het afspelen start of stopt of wordt gebufferd, of wanneer een videostream niet kan worden gevonden. De volgende code geeft de gebeurtenis `netStatusHandler()` weer:

```
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.
    }
}
```

De vorige code evalueert de eigenschap code van het Info-object en filtert wanneer de code 'NetStream.Play.Start', 'NetStream.Play.StreamNotFound' of 'NetStream.Play.Stop' is. Alle andere codes worden genegeerd. Wanneer de netwerkstream de code start, start de Timer-instantie die de afspeelkop bijwerkt. Wanneer de netwerkstream niet kan worden gevonden of is gestopt, wordt de Timer-instantie gestopt en probeert de toepassing de volgende video in de afspeellijst af te spelen.

Elke keer dat de timer wordt uitgevoerd, werkt de `positionBar`-voortgangsbalkinstantie de huidige positie bij door de methode `setProgress()` van de klasse ProgressBar op te roepen en wordt de Label-instantie `positionLabel` bijgewerkt met de verstreken tijd en de totale tijd van de huidige video.

```
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.
    }
}
```

Het volume van de video regelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt het volume van een dynamisch geladen video regelen door de eigenschap `soundTransform` op het NetStream-object in te stellen. Met de videojukeboxtoepassing kunt u het volumeniveau wijzigen door de waarde van de Slider-instantie `volumeSlider` te wijzigen. De volgende code toont hoe u het volumeniveau kunt wijzigen door de waarde van de Slider-component toe te wijzen aan een SoundTransform-object dat op de eigenschap `soundTransform` op het NetStream-object is ingesteld:

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

Video afspelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De rest van de toepassing bestuurt het afspelen van de video wanneer de video het einde van de videostream heeft bereikt of wanneer de gebruiker naar de vorige of volgende video gaat.

De volgende methode haalt de video-URL op van de XMLList voor de index die momenteel is geselecteerd:

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

De methode `playVideo()` roept de methode `play()` op het NetStream-object op om de video te laden die momenteel is geselecteerd:

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

De methode `playPreviousVideo()` verlaagt de huidige video-index, roept de methode `playVideo()` op om het nieuwe videobestand te laden en stelt de voortgangsbalk op visible in:

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

De laatste methode, `playNextVideo()`, verhoogt de video-index en roept de methode `playVideo()` op. Als de huidige video zich in de afspeellijst bevindt, wordt de methode `clear()` op het Video-object aangeroepen en wordt de eigenschap `visible` van de voortgangsbalkinstantie op `false` ingesteld:

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

De klasse StageVideo gebruiken voor presentatie met hardwareversnelling

Flash Player 10.2 of hoger

Flash Player optimaliseert de videoprestaties aan de hand van hardwareversnelling voor H.264-decodering. U kunt de prestaties nog verder verbeteren met de StageVideo-API. Met werkgebiedvideo profiteert uw toepassing optimaal van presentatie met hardwareversnelling.

De volgende runtimes ondersteunen de StageVideo-API:

- Flash Player 10.2 of hoger

Opmerking: *In Flash Player 11.4/AIR 3.4 en hoger kunt u camera-invoer gebruiken met StageVideo.*

Downloadbare broncode en aanvullende informatie over de functie voor werkgebiedvideo is beschikbaar op Aan de slag met werkgebiedvideo.

Een zelfstudie om snel met StageVideo te leren werken vindt u op Werken met werkgebiedvideo.

Hardwareversnelling met gebruik van StageVideo

Presentatie met hardwareversnelling, waarin het schalen van video, kleuromzetting en tekenen is opgenomen, verbetert de prestaties en profiteert van decodering met hardwareversnelling. Op apparaten met GPU (hardware)-versnelling kunt u een flash.media.StageVideo-object gebruiken om video direct op het apparaat te verwerken. Directe verwerking betekent dat de CPU andere taken kan uitvoeren terwijl de GPU video verwerkt. De oudere Video-klasse gebruikt daarentegen standaard softwarepresentatie. Softwarepresentatie vindt plaats in de CPU en kan een groot deel van de systeembronnen in beslag nemen.

Momenteel verschaffen nog maar weinig apparaten volledige GPU-versnelling. Met werkgebiedvideo kunnen toepassingen echter optimaal profiteren van de beschikbare hardwareversnelling.

De aanwezigheid van de klasse StageVideo betekent niet dat de klasse Video overbodig is. Samen zorgen deze twee klassen voor de beste videoweergave die gezien de apparaatbronnen op een bepaald moment mogelijk is. Uw toepassing benut hardwareversnelling door naar de desbetreffende gebeurtenissen te luisteren en naar behoefte te schakelen tussen StageVideo en Video.

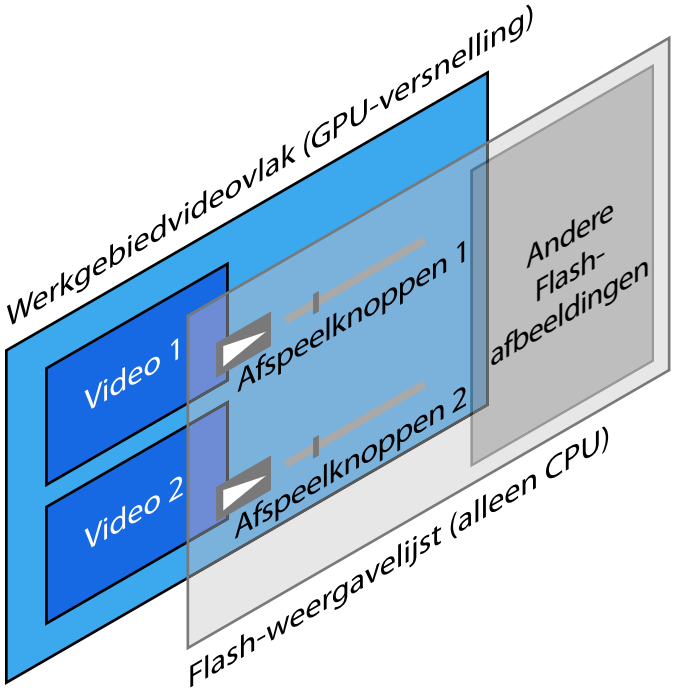
De klasse StageVideo stelt enkele beperkingen op het videoverbruik in. Lees de richtlijnen door en controleer of uw toepassing deze kan accepteren voordat u StageVideo implementeert. Als u de beperkingen accepteert, moet u de klasse StageVideo gebruiken als Flash Player detecteert dat presentatie met hardwareversnelling beschikbaar is. Zie "Richtlijnen en beperkingen" op pagina 541.

Parallelle vlakken: werkgebiedvideo en de Flash-weergavelijst

In het werkgebiedvideomodel kan Flash Player video onderscheiden van de weergavelijst. Flash Player deelt de samengestelde weergave op in twee vlakken met z-volgorde:

Werkgebiedvideovlak Het werkgebiedvideovlak bevindt zich op de achtergrond Flash Player geeft alleen video met hardwareversnelling weer. Vanwege dit ontwerp is dit vlak niet beschikbaar als hardwareversnelling niet wordt ondersteund of niet beschikbaar is op het apparaat. In ActionScript verwerken StageVideo-objecten de video's die in het werkgebiedvideovlak worden afgespeeld.

Vlak voor Flash-weergavelijst Elementen in de Flash-weergavelijst worden op een vlak vóór het werkgebiedvideovlak geplaatst. Tot deze elementen worden alle elementen gerekend die de runtime kan renderen, zoals afspeelknoppen. Als hardwareversnelling niet beschikbaar is, kunnen video's alleen in dit vlak worden afgespeeld, en wel met gebruik van het klasseobject Video. Werkgebiedvideo wordt altijd achter beelden uit de Flash-weergavelijst afgespeeld.



Videoweergavevlakken

Het StageVideo-object wordt weergegeven in een met het venster uitgelijnde, niet geroteerde rechthoek op het scherm. U kunt objecten niet in een laag achter het werkgebiedvideovlak plaatsen. U kunt het vlak met de Flash-weergavelijst echter wel gebruiken om andere beelden in een laag vóór het werkgebiedvlak te plaatsen. Werkgebiedvideo wordt tegelijkertijd met de weergavelijst uitgevoerd. U kunt de twee mechanismen dus tegelijk gebruiken om één visueel effect te maken dat gebruikmaakt van twee verschillende vlakken. U kunt het voorste vlak bijvoorbeeld gebruiken voor afspeelknoppen om de op de achtergrond uitgevoerde werkgebiedvideo te bedienen.

Werkgebiedvideo en H.264-codec

In Flash Player-toepassingen bestaat het implementeren van video met hardwareversnelling uit twee stappen:

1 De video coderen als H.264

2 De StageVideo-API implementeren

Voer beide stappen uit om de beste resultaten te bereiken. Met de H.264-codec kunt u maximaal profiteren van hardwareversnelling, van videodecodering tot presentatie.

Met werkgebiedvideo wordt het teruglezen van GPU naar CPU overbodig. Met andere woorden, de GPU hoeft gedecodeerde frames niet meer terug te sturen naar de CPU om te worden samengesteld met objecten uit de weergavelijst. De GPU stuurt gedecodeerde en gerenderde frames in plaats daarvan rechtstreeks naar het scherm, en wel achter de objecten uit de weergavelijst. Deze techniek verlaagt het CPU- en geheugenverbruik en verschaft bovendien hogere pixelprecisie.

Meer Help-onderwerpen

"Video-indelingen" op pagina 500

Richtlijnen en beperkingen

Wanneer video op het volledige scherm wordt uitgevoerd, is werkgebiedvideo altijd beschikbaar als het apparaat ondersteuning biedt voor hardwareversnelling. Flash Player wordt echter ook uitgevoerd in een browser. In een browsercontext beïnvloedt de instelling `wmode` de beschikbaarheid van werkgebiedvideo. Als vuistregel geldt dat u `wmode="direct"` altijd beter kunt gebruiken als u werkgebiedvideo wilt gebruiken. In andere modi dan de modus Volledig scherm is werkgebiedvideo niet compatibel met andere `wmode`-instellingen. Dat betekent dat werkgebiedvideo tijdens de runtime op onvoorspelbare wijze kan afwisselen tussen beschikbaar en niet beschikbaar. Als de gebruiker bijvoorbeeld de modus Volledig scherm sluit terwijl werkgebiedvideo wordt uitgevoerd, verandert de videocontext weer terug naar de browser. Als de browserparameter `wmode` niet is ingesteld op `"direct"`, kan werkgebiedvideo ineens onbeschikbaar worden. Flash Player meldt wijzigingen in de afspeelcontext aan de hand van een serie gebeurtenissen aan toepassingen. Als u de StageVideo-API implementeert, bewaart u een Video-object als back-up voor het geval werkgebiedvideo onbeschikbaar zou worden.

Vanwege het rechtstreekse verband met de hardware, beperkt werkgebiedvideo enkele videofuncties. Werkgebiedvideo dwingt de volgende beperkingen af:

- Flash Player beperkt voor elk SWF-bestand het aantal StageVideo-objecten dat tegelijkertijd video's kan weergeven tot vier. De feitelijke limiet kan echter lager zijn, afhankelijk van de hardwarebronnen van het apparaat.
- De timing van de video is niet gesynchroniseerd met de timing van de inhoud die in de runtime wordt weergegeven.
- Het weergavegebied van de video is altijd een rechthoek. U kunt geen geavanceerde weergavegebieden gebruiken, zoals een ellips of onregelmatige vormen.
- De video kan niet worden gedraaid.
- U kunt de video niet in de bitmapcache plaatsen en niet openen via het BitmapData-object.
- Er kunnen geen filters worden toegepast op de video.
- Er kunnen geen kleurtransformaties worden toegepast op de video.
- Er kan geen alphawaarde worden toegepast op de video.
- Overvloeimodi die u toepast op objecten in het weergavelijstvlak zijn niet van toepassing op werkgebiedvideo.
- De video kan alleen worden geplaatst op grenzen met hele pixels.

- Hoewel GPU-rendering de beste beschikbare rendering voor de desbetreffende hardware is, is er geen sprake van 100% identieke pixels op verschillende apparaten. Er kunnen kleine verschillen optreden die te wijten zijn aan stuurprogramma- en platformverschillen.
- Enkele apparaten bieden geen ondersteuning voor alle vereiste kleurruimten. Sommige apparaten bieden bijvoorbeeld geen ondersteuning voor BT.709, de H.264-standaard. In dergelijke gevallen kunt u BT.601 gebruiken voor snelle weergave.
- U kunt werkgebiedvideo niet gebruiken met WMODE-instellingen, zoals Normaal, Dekkend of Transparant. Werkgebiedvideo ondersteunt `WMODE=direct` alleen als de modus Volledig scherm niet actief is. WMODE heeft geen effect in Safari 4 of hoger en in IE 9 of hoger.

In de meeste gevallen beïnvloeden deze beperkingen videospelertoepassingen niet. Gebruik waar mogelijk werkgebiedvideo als u deze beperkingen kunt accepteren.

Meer Help-onderwerpen

"Werken met de modus Volledig scherm" op pagina 177

De StageVideo-API's gebruiken

Werkgebiedvideo is een runtimemechanisme dat de videoweergave en de prestaties van het apparaat verbetert. De runtime creëert en onderhoudt dit mechanisme. Uw rol als ontwikkelaar bestaat uit het configureren van uw toepassing om ervan te profiteren.

Als u werkgebiedvideo wilt gebruiken, implementeert u een framework met gebeurtenishandlers die detecteren wanneer werkgebiedvideo al dan niet beschikbaar is. Wanneer wordt gemeld dat werkgebiedvideo beschikbaar is, haalt u een StageVideo-object op van de `Stage.stageVideos`-eigenschap. De runtime vult dit Vector-object met een of meerdere StageVideo-objecten. U kunt dan een van de verschafte StageVideo-objecten gebruiken in plaats van een Video-object om streaming video weer te geven.

Wanneer in Flash Player wordt gemeld dat werkgebiedvideo niet meer beschikbaar is, schakelt u de videostream terug naar een Video-object.

Opmerking: *U kunt geen StageVideo-objecten maken.*

De eigenschap Stage.stageVideos

De eigenschap `Stage.stageVideos` is een Vector-object dat u toegang geeft tot StageVideo-instanties. Dit Vector-object kan maximaal vier StageVideo-objecten bevatten, afhankelijk van de hardware- en systeembronnen. Op mobiele apparatuur kan dit beperkt zijn tot één of zelfs geen.

Als er geen werkgebiedvideo beschikbaar is, bevat dit Vector-object geen objecten. Zorg er, om runtimefouten te voorkomen, voor dat u leden van deze vector alleen benadert als de meest recente `StageVideoAvailability`-gebeurtenis aangeeft dat werkgebiedvideo beschikbaar is.

StageVideo-gebeurtenissen

Het StageVideo API-framework biedt de volgende gebeurtenissen:

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY Wordt verzonden wanneer de eigenschap `Stage.stageVideos` verandert. De eigenschap `StageVideoAvailabilityEvent.availability` geeft `AVAILABLE` of `UNAVAILABLE` aan. Gebruik deze gebeurtenis om te bepalen of de eigenschap `stageVideos` StageVideo-objecten bevat, in plaats van rechtstreeks de lengte van de `Stage.stageVideos`-vector te controleren.

StageVideoEvent.RENDER_STATE Wordt verzonden wanneer een NetStream- of Camera-object is gekoppeld aan een StageVideo-object en wordt afgespeeld. Verwijst naar het type decodering dat momenteel wordt gebruikt: hardware,

software of niet beschikbaar (er wordt niets weergegeven). Het gebeurtenisdoel bevat de eigenschappen `videoWidth` en `videoHeight` die u kunt gebruiken voor het wijzigen van het formaat van de videoviewport.

Belangrijk: *De van het StageVideo-doelobject verkregen coördinaten gebruiken Stage-coördinaten, aangezien deze niet deel uitmaken van de standaardweergavelijst.*

VideoEvent.RENDER_STATE Wordt verzonden wanneer een Video-object wordt gebruikt. Geeft aan of softwaredecodering of decodering met hardwareversnelling wordt gebruikt. Als deze gebeurtenis versnelde decodering aangeeft, moet u, indien mogelijk, overschakelen op een StageVideo-object. Het gebeurtenisdoel Video bevat de eigenschappen `videoWidth` en `videoHeight` die u kunt gebruiken voor het wijzigen van het formaat van de videoviewport.

Workflow voor implementatie van de StageVideo-functie

Volg deze stappen op hoofdniveau om de StageVideo-functie te implementeren:

1. Luister naar de gebeurtenis `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` om te kijken wanneer de `Stage.stageVideos`-vector is gewijzigd. Zie "De gebeurtenis StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY gebruiken" op pagina 544.
2. Als door de `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-gebeurtenis wordt gerapporteerd dat werkgebiedvideo beschikbaar is, gebruikt u het Vector-object `Stage.stageVideos` in de gebeurtenishandler om een StageVideo-object te benaderen.
3. Koppel een NetStream-object met behulp van `StageVideo.attachNetStream()` of koppel een Camera-object met behulp van `StageVideo.attachCamera()`.
4. Speel de video af met `NetStream.play()`.
5. Luister naar de gebeurtenis `StageVideoEvent.RENDER_STATE` op het StageVideo-object om de afspeelstatus van de video te bepalen. Als u deze gebeurtenis ontvangt, betekent dat ook dat de eigenschappen voor de breedte en de hoogte van de video zijn geïnitialiseerd of gewijzigd. Zie "De gebeurtenissen StageVideoEvent.RENDER_STATE en VideoEvent.RENDER_STATE gebruiken" op pagina 546.
6. Luister naar de gebeurtenis `VideoEvent.RENDER_STATE` van het Video-object. Deze gebeurtenis biedt dezelfde statussen als `StageVideoEvent.RENDER_STATE`. Met deze gebeurtenis kunt u dus ook bepalen of GPU-versnelling beschikbaar is. Als u deze gebeurtenis ontvangt, betekent dat ook dat de eigenschappen voor de breedte en de hoogte van de video zijn geïnitialiseerd of gewijzigd. Zie "De gebeurtenissen StageVideoEvent.RENDER_STATE en VideoEvent.RENDER_STATE gebruiken" op pagina 546.

StageVideo-gebeurtenislisteners initialiseren

Stel uw StageVideoAvailabilityEvent- en VideoEvent-listeners tijdens de initialisatie van de toepassing in. U kunt deze listeners bijvoorbeeld initialiseren in de `flash.events.Event.ADDED_TO_STAGE`-gebeurtenishandler. Deze gebeurtenis garandeert dat uw toepassing zichtbaar is in het werkgebied.

```
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);
        //...
    }
```

De gebeurtenis StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY gebruiken

In de `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY`-handler geeft u aan of een Video- of StageVideo-object wordt gebruikt, op basis van de beschikbaarheid van StageVideo. Als de eigenschap `StageVideoAvailabilityEvent.availability` is ingesteld op `StageVideoAvailability.AVAILABLE`, gebruikt u StageVideo. In dat geval kunt u erop vertrouwen dat de Stage.stageVideos-vector een of meerdere StageVideo-objecten bevat. Haal een StageVideo-object op uit de eigenschap `Stage.stageVideos` en koppel het NetStream-object eraan. Aangezien StageVideo-objecten altijd op de achtergrond worden weergegeven, verwijdert u alle bestaande Video-objecten (die altijd op de voorgrond worden weergegeven). Benut deze gebeurtenishandler ook om een listener toe te voegen voor de `StageVideoEvent.RENDER_STATE`-gebeurtenis.

Wanneer de eigenschap `StageVideoAvailabilityEvent.availability` is ingesteld op `StageVideoAvailability.UNAVAILABLE`, gebruikt u StageVideo niet en benadert u de `Stage.stageVideos`-vector niet. In dit geval koppelt u het NetStream-object aan een Video-object. Ten slotte voegt u het StageVideo- of Video-object toe aan het werkgebied en roept u `NetStream.play()` aan.

De volgende code illustreert hoe u de gebeurtenis `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` moet afhandelen:

```
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);
    }
}
```

Belangrijk: *Wanneer een toepassing het vectorelement voor het eerst benadert op Stage.stageVideos[0], wordt het kader standaard ingesteld op 0,0,0,0 en worden standaardwaarden gebruikt voor pannen en zoomen. Stel deze waarden altijd weer in op de instellingen waaraan u de voorkeur geeft. U kunt de eigenschappen* `videoWidth` *en* `videoHeight` *van het gebeurtenisdoel* `StageVideoEvent.RENDER_STATE` *of* `VideoEvent.RENDER_STATE` *gebruiken om de afmetingen van de videoviewport te berekenen.*

U kunt de volledige broncode van deze voorbeeldtoepassing downloaden van Aan de slag met werkgebiedvideo.

De gebeurtenissen StageVideoEvent.RENDER_STATE en VideoEvent.RENDER_STATE gebruiken

StageVideo- en Video-objecten verzenden gebeurtenissen om toepassingen te melden wanneer de weergaveomgeving verandert. `StageVideoEvent.RENDER_STATE` en `VideoEvent.RENDER_STATE`.

Een StageVideo- of Video-object verzendt een renderstatusgebeurtenis wanneer een NetStream-object wordt gekoppeld en afgespeeld. Het verzendt deze gebeurtenis ook wanneer de weergaveomgeving verandert, bijvoorbeeld wanneer het formaat van de videoviewport wordt gewijzigd. Gebruik deze meldingen om uw viewport opnieuw in te stellen op de huidige waarden voor `videoHeight` en `videoWidth` van het gebeurtenisdoelobject.

Tot de gemelde renderstatussen behoren:

- `RENDER_STATUS_UNAVAILABLE`
- `RENDER_STATUS_SOFTWARE`
- `RENDER_STATUS_ACCELERATED`

Renderstatussen geven aan wanneer decodering met hardwareversnelling wordt gebruikt, ongeacht de klasse die de video momenteel afspeelt. Controleer de eigenschap `StageVideoEvent.status` om te zien of de vereiste decodering beschikbaar is. Wanneer deze eigenschap is ingesteld op "unavailable", kan het StageVideo-object de video niet afspelen. Deze status vereist dat u het NetStream-object meteen weer aan een Video-object koppelt. Andere statussen stellen uw toepassing op de hoogte van de huidige renderomstandigheden.

In de volgende tabel worden de implicaties beschreven van alle renderstatuswaarden voor StageVideoEvent- en VideoEvent-objecten in Flash Player:

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	Decodering en presentatie vinden beide plaats in de hardware. (Optimale prestaties.)	Presentatie in hardware, decodering in software. (Acceptabele prestaties.)	Er zijn geen GPU-bronnen beschikbaar om de video te verwerken en er wordt niets weergegeven. **Fallback naar een Video-object.**
VideoEvent	Presentatie in software, decodering in hardware. (Alleen acceptabele prestaties op een modern desktopsysteem. Mindere prestaties op volledig scherm.)	Presentatie in software, decodering in software. (Minst goede prestaties. Mindere prestaties op volledig scherm.)	(N.v.t.)

Kleurruimten

Werkgebiedvideo gebruikt de onderliggende hardwarefuncties om kleurruimten te ondersteunen. SWF-inhoud kan metagegevens verschaffen die de kleurruimte die de voorkeur geniet, aangeven. De grafische hardware bepaalt echter of de kleurruimte kan worden gebruikt. Bepaalde apparaten bieden ondersteuning voor verschillende kleurruimten, terwijl andere apparaten geen enkele kleurruimte ondersteunen. Als de hardware geen ondersteuning biedt voor de gewenste kleurruimte, probeert Flash Player de meest overeenkomende kleurruimte te vinden die wel wordt ondersteund.

Gebruik de eigenschap `StageVideo.colorSpaces` om te zien welke kleurruimten door de hardware worden ondersteund. Deze eigenschap retourneert de lijst met ondersteunde kleurruimten in een String-vector:

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

Controleer de eigenschap `StageVideoEvent.colorSpace` om te zien welke kleurruimte de momenteel afgespeelde video gebruikt. Controleer deze eigenschap in de gebeurtenishandler voor de gebeurtenis `StageVideoEvent.RENDER_STATE`:

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Als Flash Player geen vervanging kan vinden voor een niet-ondersteunde kleurruimte, gebruikt werkgebiedvideo de standaardkleurruimte BT.601. Videostreams met H.264-codering maken bijvoorbeeld doorgaans gebruik van de BT.709-kleurruimte. Als de hardware geen ondersteuning biedt voor BT.709, retourneert de eigenschap `colorSpace` de waarde `"BT601"`. De `StageVideoEvent.colorSpace`-waarde `"unknown"` geeft aan dat de hardware de kleurruimte niet kan opvragen.

Als uw toepassing de actieve kleurruimte niet acceptabel acht, kunt u van een StageVideo-object overschakelen op een Video-object. De Video-klasse ondersteunt alle kleurruimten via softwaresamenstelling.

Hoofdstuk 26: Werken met camera's

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een camera die op de computer van de gebruiker is aangesloten, kan fungeren als bron van videogegevens die u met ActionScript kunt weergeven en manipuleren. De klasse Camera is het in ActionScript geïntegreerde mechanisme voor het werken met de camera van een computer of apparaat.

Op mobiele apparatuur kunt ook de klasse CameraUI gebruiken. De klasse CameraUI start een afzonderlijke cameratoepassing waarmee de gebruiker een foto of video kan vastleggen. Wanneer de gebruiker klaar is, kan de toepassing de afbeelding of de video benaderen door middel van een MediaPromise-object.

Meer Help-onderwerpen

Christian Cantrell: How to use CameraUI in a Cross-platform Way

Michaël CHAIZE: Android, AIR and the Camera

De klasse Camera

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met het Camera-object kunt u verbinding maken met de lokale camera van de gebruiker en de video lokaal (terug naar de gebruiker) of extern naar de server (zoals Flash Media Server) uitzenden.

Wanneer u de klasse Camera gebruikt, hebt u toegang tot de volgende soorten informatie over de camera van de gebruiker:

- Welke op de computer of het apparaat van de gebruiker geïnstalleerde camera's beschikbaar zijn
- Of een camera is geïnstalleerd
- Of Flash Player toegang heeft tot de camera van de gebruiker
- Welke camera op dit moment actief is
- De breedte en hoogte van de video die wordt vastgelegd

De klasse Camera bevat verschillende nuttige methoden en eigenschappen die bij het werken met Camera-objecten kunnen worden gebruikt. De statische eigenschap `Camera.names` bevat bijvoorbeeld een array van namen van camera's die op dit moment op de computer van de gebruiker zijn geïnstalleerd. U kunt ook de eigenschap `name` gebruiken om de naam weer te geven van de camera die momenteel actief is.

Opmerking: *Tijdens het streamen van videobeelden van een camera via het netwerk, dient u netwerkonderbrekingen altijd af te handelen. Er kunnen vele redenen zijn voor netwerkonderbrekingen, vooral op mobiele apparatuur.*

Camera-inhoud op het scherm weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u verbinding maakt met een camera, hoeft u minder code te schrijven dan wanneer u de klassen NetConnection en NetStream gebruikt om een videobestand te laden. Het gebruik van de klasse Camera kan ook vaak lastig zijn omdat u met Flash Player toestemming van de gebruiker nodig hebt om verbinding te maken met de camera voordat u toegang tot de camera kunt krijgen.

De volgende code geeft aan hoe u de klasse Camera kunt gebruiken om verbinding te maken met de lokale camera van de gebruiker:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Opmerking: *De klasse Camera bevat geen constructormethode. Wanneer u een nieuwe Camera-instantie wilt maken, gebruikt u de statische methode* `Camera.getCamera()`.

Cameratoepassingen ontwerpen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een toepassing schrijft die verbinding maakt met de camera van de gebruiker, moet u in uw code met het volgende rekening houden:

- Controleer of de gebruiker een camera heeft geïnstalleerd. Handel het geval af als geen camera beschikbaar is.
- Alleen voor Flash Player: controleer of de gebruiker expliciet toegang tot de camera heeft toegestaan. De speler geeft uit veiligheidsoverwegingen het dialoogvenster Instellingen Flash Player weer, waarmee de gebruiker toegang tot de camera kan toestaan of weigeren. Hierdoor wordt voorkomen dat Flash Player verbinding maakt met een camera van de gebruiker en een videostream uitzendt zonder dat hiervoor toestemming is gegeven. Wanneer de gebruiker op Toestaan klikt, mag uw toepassing verbinding maken met de camera van de gebruiker. Wanneer de gebruiker op Weigeren klikt, heeft uw toepassing geen toegang tot de camera van de gebruiker. In uw toepassingen moeten beide gevallen altijd netjes worden afgehandeld.
- Alleen AIR: controleer of de Camera-klasse wordt ondersteund voor de apparaatprofielen die door uw toepassing worden ondersteund.
- De Camera-klasse wordt niet ondersteund in mobiele browsers.
- De Camera-klasse wordt niet ondersteund in mobiele AIR-toepassingen die de GPU-rendermodus gebruiken.
- Op mobiele apparaten kan slechts één camera tegelijkertijd actief zijn.

Meer Help-onderwerpen

Christophe Coenraets: Multi-User Video Tic-Tac-Toe

Mark Doherty: Android Radar app (source)

Verbinding maken met de camera van een gebruiker

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij de eerste stap voor het maken van verbinding met de camera van de gebruiker, maakt u een nieuwe Camera-instantie door een variabele van het type Camera te maken en deze te initialiseren op de geretourneerde waarde van de statische methode `Camera.getCamera()`.

Bij de tweede stap moet u een nieuw Video-object maken en het Camera-object hieraan koppelen.

Bij de derde stap voegt u het Video-object aan de weergavelijst toe. De stappen 2 en 3 moeten worden uitgevoerd omdat de klasse Camera de klasse DisplayObject niet uitbreidt. Hierdoor kan deze niet direct aan de weergavelijst worden toegevoegd. Wanneer u de vastgelegde video van de camera wilt afspelen, maakt u een nieuw Video-object en roept u de methode `attachCamera()` op.

De volgende code geeft deze drie stappen weer:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Als de gebruiker geen camera heeft geïnstalleerd, geeft de toepassing niets weer.

In de praktijk moet u mogelijk aanvullende stappen aan uw toepassing toevoegen. Zie "Controleren of camera's zijn geïnstalleerd" op pagina 550 en "Bevoegdheden detecteren voor cameratoegang" op pagina 551 voor meer informatie.

Meer Help-onderwerpen

Lee Brimelow: How to access the camera on Android devices

Controleren of camera's zijn geïnstalleerd

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voordat u probeert methoden of eigenschappen voor een Camera-instantie te gebruiken, controleert u of de gebruiker een camera heeft geïnstalleerd. Er zijn twee manieren om te controleren of de gebruiker een camera heeft geïnstalleerd:

- Controleer de statische eigenschap `Camera.names`, die een array bevat van namen van camera's die beschikbaar zijn. Deze array bevat normaal gesproken één of minder tekenreeksen, omdat de meeste gebruikers waarschijnlijk niet meer dan één camera tegelijk hebben geïnstalleerd. De volgende code toont hoe u de eigenschap `Camera.names` kunt controleren om te bepalen of de gebruiker beschikbare camera's heeft:

```
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.");
}
```

- Controleer de geretourneerde waarde van de statische methode `Camera.getCamera()`. Wanneer er geen camera's beschikbaar of geïnstalleerd zijn, retourneert deze methode `null`; anders retourneert de methode een verwijzing naar het Camera-object. De volgende code toont hoe u de methode `Camera.getCamera()` kunt controleren om te bepalen of de gebruiker beschikbare camera's heeft:

```
var cam:Camera = Camera.getCamera();
if (cam == null)
{
    trace("User has no cameras installed.");
}
else
{
    trace("User has at least 1 camera installed.");
}
```

Omdat de klasse Camera de klasse DisplayObject niet uitbreidt, kan deze niet direct aan de weergavelijst worden toegevoegd met de methode `addChild()`. Wanneer u de vastgelegde video van de camera wilt afspelen, moet u een nieuw Video-object maken en de methode `attachCamera()` op de Video-instantie aanroepen.

In dit fragment wordt weergegeven hoe u de camera kunt aansluiten, als die bestaat. Als dat niet het geval is, geeft de toepassing niets weer:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

Camera's op mobiele apparatuur

De klasse Camera wordt in mobiele browsers niet ondersteund in de Flash Player-runtime.

In AIR-toepassingen op mobiele apparatuur kunt u de camera of camera's op het apparaat benaderen. Op mobiele apparaten kunt u zowel de naar voren als de naar achteren gerichte camera gebruiken, maar de uitvoer van slechts één camera kan tegelijkertijd worden weergegeven. (Wanneer u een tweede camera aansluit, wordt de eerste losgekoppeld.) De naar voren gerichte camera wordt horizontaal gespiegeld op iOs, maar op Android is dit niet het geval.

Bevoegdheden detecteren voor cameratoegang

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de AIR-toepassingssandbox kan de toepassing zonder toestemming van de gebruiker toegang krijgen tot elke camera. Op Android moet de toepassing echter de Android CAMERA-bevoegdheid opgeven in het descriptorbestand van de toepassing.

Voordat Flash Player de uitvoer van een camera kan weergeven, moet de gebruiker Flash Player expliciet toegang tot de camera geven. Wanneer de methode `attachCamera()` wordt aangeroepen, geeft Flash Player het dialoogvenster Instellingen Flash Player weer, waarin de gebruiker wordt gevraagd Flash Player toegang tot de camera en de microfoon te verlenen of te weigeren. Als de gebruiker op de knop Toestaan klikt, geeft Flash Player de uitvoer van de camera weer in de Video-instantie op het werkgebied. Wanneer de gebruiker op de knop Weigeren klikt, heeft Flash Player geen toegang tot de camera en geeft het Video-object niets weer.

Wanneer u wilt controleren of de gebruiker Flash Player toegang tot de camera heeft verleend, kunt u naar de gebeurtenis `status` (`StatusEvent.STATUS`) van de camera luisteren. Dit wordt in de volgende code geïllustreerd:

```
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"
}
```

De functie `statusHandler()` wordt aangeroepen zodra de gebruiker op Toestaan of Weigeren klikt. U kunt controleren op welke knop de gebruiker heeft geklikt door een van de volgende twee methoden te gebruiken:

- De parameter `event` van de functie `statusHandler()` bevat de eigenschap code, die de tekenreeks 'Camera.Muted' of 'Camera.Unmuted' bevat. Wanneer de waarde 'Camera.Muted' is, heeft de gebruiker op de knop Weigeren geklikt en heeft Flash Player geen toegang tot de camera. U ziet een voorbeeld in het volgende fragment:

  ```
  function statusHandler(event:StatusEvent):void
  {
      switch (event.code)
      {
          case "Camera.Muted":
              trace("User clicked Deny.");
              break;
          case "Camera.Unmuted":
              trace("User clicked Accept.");
              break;
      }
  }
  ```

- De klasse Camera bevat de alleen-lezen eigenschap `muted`, die aangeeft of de gebruiker cameratoegang heeft geweigerd (`true`) of heeft toegestaan (`false`) via het deelvenster Privacy van Flash Player. U ziet een voorbeeld in het volgende fragment:

  ```
  function statusHandler(event:StatusEvent):void
  {
      if (cam.muted)
      {
          trace("User clicked Deny.");
      }
      else
      {
          trace("User clicked Accept.");
      }
  }
  ```

Door de statusgebeurtenis te controleren die wordt verzonden, kunt u code schrijven die het toestaan of weigeren van cameratoegang door de gebruiker afhandelt en vervolgens systeemresources opschoont. Als de gebruiker bijvoorbeeld op de knop Weigeren klikt, kunt u een bericht aan de gebruiker weergeven waarin wordt aangegeven dat op de knop Toestaan moet worden geklikt als de gebruiker aan een videochat wil deelnemen. U kunt er ook voor zorgen dat het Video-object uit de weergavelijst wordt verwijderd zodat systeemresources kunnen worden opgeschoond.

In AIR verzendt een Camera-object geen statusgebeurtenissen, aangezien de bevoegdheid om de camera te gebruiken niet dynamisch is.

Videokwaliteit van de camera optimaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Nieuwe instanties van de klasse Video zijn standaard 320 pixels breed en 240 pixels hoog. Wanneer u de videokwaliteit wilt optimaliseren, moet u er altijd voor zorgen dat het Video-object dezelfde afmetingen heeft als de video die door het Camera-object wordt geretourneerd. U kunt de breedte en hoogte van het Camera-object ophalen met de eigenschappen `width` en `height` van de klasse Camera. Vervolgens kunt u de eigenschappen `width` en `height` van het Video-object instellen op de afmetingen van de Camera-objecten, zodat deze overeenkomen met die van de Camera-objecten. Ten slotte kunt u de breedte en hoogte van de camera doorgeven aan de constructormethode van de klasse Video. Dit wordt in het volgende fragment geïllustreerd:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

Aangezien de methode `getCamera()` een verwijzing aan een Camera-object doorgeeft (of `null` als er geen camera's beschikbaar zijn), hebt u toegang tot de methoden en eigenschappen van de camera, zelfs als de gebruiker cameratoegang weigert. Hierdoor kunt u de grootte van de Video-instantie met de native hoogte en breedte van de camera instellen.

```
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);
}
```

Zie de sectie over de modus Volledig scherm onder "De eigenschappen van Stage instellen" op pagina 175 voor informatie over de modus Volledig scherm.

Toezicht houden op de status van de camera

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Camera bevat verschillende eigenschappen waarmee u de huidige status van het Camera-object kunt controleren. De volgende code geeft bijvoorbeeld verschillende eigenschappen van de camera in de weergavelijst weer met een Timer-object en een tekstveldinstantie:

```
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");
}
```

Elke 1/10 van een seconde (100 milliseconden) wordt de gebeurtenis `timer` van het Timer-object verzonden en wordt het tekstveld door de functie `timerHandler()` in de weergavelijst bijgewerkt.

Hoofdstuk 27: Digitaal rechtenbeheer gebruiken

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

Adobe® Access™ is een oplossing voor inhoudsbeveiliging. Dit programma realiseert nieuwe inkomstenbronnen voor eigenaars van inhoud, distributeurs en adverteerders, door naadloos toegang te bieden tot premiuminhoud. Uitgevers gebruiken Adobe Access om inhoud te coderen, beleidsregels in te stellen en om licenties te verlenen. Adobe Flash Player en Adobe AIR beschikken over een DRM-bibliotheek, de Adobe Access-module. Hiermee kan het runtimeprogramma communiceren met de Adobe Access-licentieserver en beveiligde inhoud afspelen. Het runtimeprogramma voltooit derhalve de levenscyclus van de inhoud, die wordt beveiligd door Adobe Access en gedistribueerd door Flash Media Server.

Met Adobe Access kunnen inhoudsproviders zowel gratis inhoud als premiuminhoud leveren. Een consument wil bijvoorbeeld een tv-programma bekijken zonder advertenties. De consument meldt zich aan en betaalt de uitgever van de inhoud. Vervolgens heeft de consument met zijn verificatiegegevens toegang tot de inhoud en kan hij het programma zonder advertenties bekijken.

In een ander voorbeeld wil een consument inhoud offline bekijken wanneer hij onderweg is en geen toegang heeft tot internet. Dit type offline workflow wordt ondersteund door AIR-toepassingen. Nadat de gebruiker zich heeft geregistreerd en de uitgever voor de inhoud heeft betaald, heeft hij toegang tot de inhoud en de bijbehorende AIR-toepassing en kan hij deze downloaden van de website van de uitgever. Met de AIR-toepassing kan de gebruiker de inhoud offline bekijken tijdens de toegestane periode. Bovendien kan de inhoud worden uitgewisseld met andere apparaten in dezelfde apparaatgroep die gebruikmaken van domeinen (**Nieuw in 3.0**).

Adobe Access biedt ook ondersteuning voor anonieme toegang waarvoor geen gebruikersverificatie wordt vereist. Een uitgever kan anonieme toegang bijvoorbeeld gebruiken voor het distribueren van inhoud met advertenties. Ook kan een uitgever anonieme toegang gebruiken om voor een bepaald aantal dagen gratis toegang tot de huidige inhoud toe te staan. De leverancier van de inhoud kan ook het type en de versie van de speler opgeven die voor het afspelen van de inhoud vereist is en beperkingen op het type en de versie instellen.

Wanneer een gebruiker beveiligde inhoud wil afspelen in Flash Player of Adobe AIR, moeten de DRM API's door de toepassing worden aangeroepen. Met de DRM API's wordt de workflow voor het afspelen van de beveiligde inhoud geïnitieerd. Het runtimeprogramma neemt contact op met de licentieserver. Dit gebeurt via de Adobe Access-module. De licentieserver verifieert de gebruiker, indien nodig, en verstrekt een licentie waarmee de beveiligde inhoud kan worden afgespeeld. Het runtimeprogramma ontvangt de licentie en decodeert de inhoud, zodat deze kan worden afgespeeld.

In het volgende gedeelte wordt beschreven hoe u de toepassing kunt configureren voor het afspelen van inhoud die is beveiligd door Adobe Access. Het is niet noodzakelijk dat u precies begrijpt hoe inhoud wordt gecodeerd of hoe beleidsregels worden onderhouden in Adobe Access. Wel wordt aangenomen dat u communiceert met de Adobe Access-licentieserver voor gebruikersverificatie en voor het ophalen van de licentie. Verder wordt aangenomen dat u een toepassing ontwerpt waarop u inhoud die is beveiligd door Adobe Access, wilt afspelen.

Een overzicht van het gebruik van Adobe Access, inclusief het creëren van een beleid, vindt u in de documentatie bij Flash Access.

Meer Help-onderwerpen

flash.net.drm package

flash.net.NetConnection

flash.net.NetStream

Workflow voor beveiligde inhoud

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Belangrijk: Flash Player 11.5 en hoger integreert de Adobe Access-module, zodat een update (het aanroepen van `SystemUpdater.update(SystemUpdaterType.DRM)`) onnodig is. Dit omvat de volgende browsers en platforms:

- Flash Player 11.5 ActiveX-besturingselement voor alle platforms behalve Internet Explorer op Windows 8 op Intel-processoren
- Flash Player 11.5-insteekmodule voor alle browsers
- Adobe AIR (voor desktop en mobiel)

Dit betekent dat de update nog *steeds vereist* is in de volgende gevallen:

- Internet Explorer op Windows 8 op Intel-processoren
- Flash Player 11.4 en lager, behalve op Google Chrome 22 en hoger (alle platforms) of 21 en hoger (Windows)

Opmerking: *U kunt `SystemUpdater.update(SystemUpdaterType.DRM)` nog steeds veilig aanroepen op een systeem met Flash Player 11.5 of hoger, maar er wordt niet gedownload.*

In de volgende overzichtsworkflow kunt u zien hoe u met een toepassing beveiligde inhoud kunt ophalen en afspelen. Er wordt aangenomen dat de toepassing specifiek is ontworpen voor het afspelen van inhoud die is beveiligd door Adobe Access:

1. Haal de metagegevens van de inhoud op.
2. Handel indien nodig updates voor Flash Player af.
3. Controleer of er een lokale licentie is. Indien wel, laad de licentie en ga naar stap 7. Indien niet, ga naar stap 4.
4. Controleer of verificatie wordt vereist. Indien niet, ga naar stap 7.
5. Indien verificatie wordt vereist, verkrijg de verificatiegegevens van de gebruiker en geef deze door aan de licentieserver.
6. Als domeinregistratie is vereist, voegt u zich bij het domein (AIR 3.0 en hoger).
7. Na succesvolle verificatie kan de licentie worden gedownload van de server.
8. Speel de inhoud af.

Als er geen fout is opgetreden en de gebruiker is geautoriseerd om de inhoud weer te geven, verzendt het NetStream-object een DRMStatusEvent-object. Hierna begint de toepassing met het afspelen. Het DRMStatusEvent-object bevat de gerelateerde bongegevens die het beleid en de toegangsrechten voor de gebruiker identificeren. In object vindt u bijvoorbeeld informatie over het feit of de inhoud offline beschikbaar kan worden gemaakt of wanneer de licentie verloopt. De toepassing kan deze gegevens gebruiken om de gebruiker zijn status mee te delen. Voorbeeld: de toepassing kan in een statusbalk aangeven hoeveel dagen de gebruiker de inhoud nog kan bekijken.

Als offlinetoegang is toegestaan, wordt de voucher in het cachegeheugen opgeslagen en wordt de gecodeerde inhoud gedownload naar de computer van de gebruiker. De inhoud is beschikbaar gedurende de periode die is gedefinieerd in de duur van de licentiecache. De eigenschap `detail in de gebeurtenis bevat "DRM.voucherObtained"`. De toepassing bepaalt waar de inhoud lokaal wordt opgeslagen om deze vervolgens offline toegankelijk te maken. U kunt vouchers ook vooraf laden met de klasse DRMManager.

Opmerking: *Het opslaan in de cache en het vooraf laden van vouchers is zowel toegestaan in AIR als in Flash Player. Het downloaden en opslaan van gecodeerde inhoud wordt echter alleen ondersteund in AIR.*

Het is de verantwoordelijkheid van de toepassing om de foutgebeurtenissen expliciet af te handelen. Bijvoorbeeld als de gebruiker geldige gegevens invoert maar de voucher die de gecodeerde inhoud beveiligt, de toegang tot de inhoud beperkt. Zo heeft een geverifieerde gebruiker geen toegang tot de inhoud als er niet voor de rechten is betaald. Hetzelfde geval kan zich tevens voordoen wanneer twee geregistreerde leden van dezelfde uitgever inhoud proberen te delen, waarvoor slechts één van de leden heeft betaald. De toepassing moet de gebruiker op de hoogte stellen van de fout en een alternatieve suggestie bieden. De alternatieve suggestie zou in dit geval kunnen bestaan uit instructies voor het registreren en betalen voor weergaverechten.

Gedetailleerde API-workflow

Flash Player 10.1 of hoger, AIR 2.0 of hoger

Hier vindt u een meer gedetailleerde versie van de workflow voor beveiligde inhoud. In deze workflow worden de specifieke API's beschreven waarmee de inhoud die is beveiligd door Adobe Access wordt afgespeeld.

1 Gebruik een URLLoader-object om de bytes van het metagegevensbestand van de beveiligde inhoud te laden. Stel dit object in als een variabele, bijvoorbeeld `metadata_bytes`.

   Alle inhoud die door Adobe Access wordt beheerd, beschikt over Adobe Access-metagegevens. Wanneer de inhoud in een pakket wordt geplaatst, kunnen deze metagegevens worden opgeslagen als een afzonderlijk metagegevensbestand (.metadata), naast de inhoud. Raadpleeg de documentatie bij Adobe Access voor meer informatie.

2 Creëer een DRMContentData-instantie. Plaats deze code in een 'try-catch'-blok:

   ```
   new DRMContentData(metadata_bytes)
   ```

   waarbij *`metadata_bytes`* staat voor het URLLoader-object van stap 1.

3 (Alleen Flash Player) Het runtimeprogramma zoekt naar de Adobe Access-module. Als deze niet wordt gevonden, wordt een IllegalOperationError gegenereerd met DRMErrorEvent-foutcode 3344 of DRMErrorEvent-foutcode 3343.

   Download de Adobe Access-module met de SystemUpdater-API om deze fout af te handelen. Nadat u deze module hebt gedownload, verzendt het SystemUpdater-object een COMPLETE-gebeurtenis. Neem een gebeurtenislistener op voor deze gebeurtenis die teruggaat naar stap 2 wanneer deze gebeurtenis wordt verzonden. Deze stappen worden in de volgende code gedemonstreerd:

   ```
   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);
   }
   ```

   Als de versie van Player zelf moet worden bijgewerkt, wordt een statusgebeurtenis verzonden. Zie “Luisteren naar een updategebeurtenis” op pagina 574 voor meer informatie over het afhandelen van deze gebeurtenis.

Opmerking: *Bij AIR-toepassingen wordt het bijwerken van de Adobe Access-module en de vereiste runtime-updates afgehandeld door het installatieprogramma van AIR.*

4 Maak listeners om te luisteren naar DRMStatusEvent en DRMErrorEvent die worden verzonden door het DRMManager-object:

```
DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
```

Controleer in de DRMStatusEvent-listener dat de voucher geldig is (niet null). Verwerk DRMErrorEvents in de DRMErrorEvent-listener. Zie "De klasse DRMStatusEvent gebruiken" op pagina 566 en "De klasse DRMErrorEvent gebruiken" op pagina 571.

5 Laad de voucher (licentie) die nodig is om de inhoud af te spelen.

Probeer eerst om de inhoud af te spelen door een lokaal opgeslagen licentie te laden.

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

Na het laden wordt `DRMStatusEvent.DRM_Status` verzonden door het DRMManager-object.

6 Als het DRMVoucher-object niet de waarde null heeft, is de voucher geldig. Ga verder met stap 13.

7 Als het DRMVoucher-object de waarde null heeft, controleert u de verificatiemethode die volgens de beleidsregels vereist is voor deze inhoud. Gebruik de eigenschap `DRMContentData.authenticationMethod`.

8 Als de verificatiemethode `ANONYMOUS` is, gaat u naar stap 13.

9 Als de verificatiemethode `USERNAME_AND_PASSWORD` is, moet uw toepassing een mechanisme bieden waarmee de gebruiker zijn verificatiegegevens kan invoeren. Geef deze verificatiegegevens door aan de licentieserver, zodat de gebruiker kan worden geverifieerd.

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

De DRMManager verzendt een `DRMAuthenticationErrorEvent` als de verificatie mislukt en een `DRMAuthenticationCompleteEvent` als de verificatie lukt. Maak listeners voor deze gebeurtenissen.

10 Als de verificatiemethode `UNKNOWN` is, dient een aangepaste verificatiemethode te worden gebruikt. In dit geval heeft de inhoudsprovider ervoor gekozen de verificatie offline uit te voeren door niet gebruik te maken van de ActionScript 3.0-API's. De aangepaste verificatieprocedure dient een verificatietoken op te leveren dat kan worden doorgegeven aan de `DRMManager.setAuthenticationToken()`-methode.

11 Als de verificatie mislukt, moet uw toepassing teruggaan naar stap 9. Zorg ervoor dat uw toepassing een mechanisme heeft voor het omgaan met en beperken van herhaalde mislukte verificatiepogingen. Zo kunt u na drie pogingen een bericht weergeven voor de gebruiker dat de verificatie is mislukt en dat de inhoud niet kan worden afgespeeld.

12 Als u het opgeslagen token wilt gebruiken, in plaats van de gebruiker te vragen naar zijn verificatiegegevens, kunt u het token instellen met de methode `DRMManager.setAuthenticationToken()`. Vervolgens downloadt u de licentie van de licentieserver en speelt u de inhoud af, zoals aangegeven in stap 8.

13 Als de verificatie is gelukt, kunt u het verificatietoken vastleggen (optioneel). Dit token is een bytearray die in het geheugen is opgeslagen. U kunt het token ophalen met de eigenschap `DRMAuthenticationCompleteEvent.token`. Vervolgens kunt u het verificatietoken opslaan en toepassen, zodat de gebruiker niet telkens zijn verificatiegegevens voor de desbetreffende inhoud hoeft in te voeren. De licentieserver bepaalt de geldige periode van het verificatietoken.

14 Na succesvolle verificatie kan de licentie worden gedownload van de licentieserver:

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

Na het laden wordt DRMStatusEvent.DRM_STATUS verzonden door het DRMManager-object. Luister naar deze gebeurtenis. Wanneer de gebeurtenis wordt verzonden kunt u de inhoud afspelen.

15 Speel de video af door een NetStream-object te maken en vervolgens de methode `play()` hiervan aan te roepen.

```
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);
```

DRMContentData- en sessieobjecten

Wanneer `DRMContentData` wordt gemaakt, wordt deze gebruikt als een sessieobject dat naar de Flash Player DRM-module verwijst. Alle `DRMManager`-API's die deze `DRMContentData` ontvangen, gebruiken deze specifieke DRM-module. Er zijn echter twee `DRMManager`-API's die niet gebruikmaken van `DRMContentData`, namelijk:

1. `authenticate()`
2. `setAuthenticationToken()`

Aangezien er geen gekoppelde `DRMContentData` zijn, wordt na het aanroepen van `DRMManager`-API's de meest recente DRM-module van de schijf gebruikt. Dit kan problemen opleveren als de DRM-module wordt bijgewerkt tijdens de DRM-workflow van de toepassing. Overweeg het volgende scenario:

1. De toepassing maakt een `DRMContentData`-object `contentData1` dat *AdobeCP1* gebruikt als de DRM-module.
2. De toepassing roept de methode `DRMManager.authenticate(contentData1.serverURL,...)` aan.
3. De toepassing roept de methode `DRMManager.loadVoucher(contentData1, ...)` aan.

Als een update van de DRM-module wordt uitgevoerd voordat de toepassing stap 2 bereikt, voert de methode `DRMManager.authenticate()` de verificatie uit met gebruik van *AdobeCP2* als de DRM-module. De `loadVoucher()`-methode in stap 3 mislukt, aangezien deze nog steeds *AdobeCP1* gebruikt als de DRM-module. De update is wellicht uitgevoerd omdat een andere toepassing de update van de DRM-module heeft aangeroepen. U kunt dit scenario vermijden door de update van de DRM-module tijdens het starten van de toepassing aan te roepen.

DRM-gerelateerde gebeurtenissen

Wanneer een toepassing beveiligde inhoud wil afspelen, verzendt het runtimeprogramma een groot aantal gebeurtenissen:

- DRMDeviceGroupErrorEvent (alleen AIR), verzonden door DRMManager
- DRMAuthenticateEvent (alleen AIR), verzonden door NetStream
- DRMAuthenticationCompleteEvent, verzonden door DRMManager
- DRMAuthenticationErrorEvent, verzonden door DRMManager
- DRMErrorEvent, verzonden door NetStream en DRMManager
- DRMStatusEvent, verzonden door NetStream en DRMManager
- StatusEvent
- NetStatusEvent. Zie “Luisteren naar een updategebeurtenis” op pagina 574.

Als u inhoud wilt ondersteunen die wordt beveiligd door Adobe Access, moet u gebeurtenislisteners toevoegen om de DRM-gebeurtenissen af te handelen.

Vouchers voor offline afspelen vooraf laden

Adobe AIR 1.5 of hoger

U kunt de vouchers (licenties) die nodig zijn om de door Adobe Access beveiligde inhoud af te spelen vooraf laden. Als gebruikers beschikken over vooraf geladen vouchers, kunnen ze de inhoud ook weergeven wanneer ze geen actieve internetverbinding hebben. (Voor het vooraf laden is wel een internetverbinding vereist.) Met de methode `preloadEmbeddedMetadata()` van de NetStream-klasse en de DRMManager-klasse kunt u vouchers op voorhand laden. Bij AIR 2.0 en hoger kunt u vouchers nu rechtstreeks vooraf laden met een DRMContentData-object. Deze techniek verdient de voorkeur, omdat u het DRMContentData-object onafhankelijk van de inhoud kunt bijwerken. (Met de methode `preloadEmbeddedData()` wordt DRMContentData opgehaald van de inhoud.)

DRMContentData gebruiken

Adobe AIR 2.0 of hoger

De volgende procedure beschrijft de workflow voor het vooraf laden van de voucher voor een beveiligd mediabestand met een DRMContentData-object.

1. Haal de binaire metagegevens op voor de inhoud die in een pakket is opgenomen. Als u de Access Java Reference Packager gebruikt, wordt dit metagegevensbetand automatisch gegenereerd. Het bestand heeft de extensie *.metadata*. U kunt deze metagegevens bijvoorbeeld downloaden met de URLLoader-klasse.
2. Maak een DRMContentData-object en geef de metagegevens door aan de constructorfunctie:

   ```
   var drmData:DRMContentData = new DRMContentData( metadata );
   ```

3. De overige stappen zijn gelijk aan de procedure die is beschreven in "Workflow voor beveiligde inhoud" op pagina 557.

preloadEmbeddedMetadata() gebruiken

Adobe AIR 1.5 of hoger

In de volgende stappen wordt de workflow beschreven voor het vooraf laden van een voucher voor een DRM-beveiligd mediabestand met `preloadEmbeddedMetadata()`:

1. Download het mediabestand en sla het op. (DRM-metagegevens kunnen alleen vooraf worden geladen uit lokaal opgeslagen bestanden.)
2. Maak de NetConnection- en NetStream-objecten en geef hierbij implementaties op voor de callback-functies `onDRMContentData()` en `onPlayStatus()` van het NetStream-clientobject.
3. Maak een NetStreamPlayOptions-object en stel de waarde van de eigenschap `stream` in op de URL van het lokale mediabestand.
4. Roep de methode `preloadEmbeddedMetadata()` van NetStream aan en geef het NetStreamPlayOptions-object door waarin het mediabestand wordt geïdentificeerd dat moet worden geparseerd.
5. Als het mediabestand DRM-metagegevens bevat, wordt de callback-functie `onDRMContentData()` aangeroepen. De metagegevens worden doorgegeven naar deze functie in de vorm van een DRMContentData-object.
6. Gebruik het DRMContentData-object om de bon te verkrijgen met behulp van de methode `loadVoucher()` van DRMManager.

Als de eigenschap `authenticationMethod` van het DRMContentData-object de waarde `flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD` heeft, moet u de gebruiker verifiëren op de mediarechtenserver voordat u de voucher laadt. De eigenschappen `serverURL` en `domain` van het DRMContentData-object kunnen samen met de gebruikersreferenties worden doorgegeven aan de DRMManager-methode `authenticate()`.

7 De callback-functie `onPlayStatus()` wordt geactiveerd wanneer de bestandsparsering is voltooid. Als de functie `onDRMContentData()` niet is aangeroepen, bevat het bestand niet de vereiste metagegevens om een voucher te verkrijgen. Als deze aanroep ontbreekt, betekent dit mogelijk ook dat dit bestand niet door Adobe Access wordt beveiligd.

In de volgende code voor AIR wordt geïllustreerd hoe u een voucher vooraf kunt laden voor een lokaal mediabestand:

```
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();
    }
}
}
```

Leden en gebeurtenissen van de klasse NetStream die betrekking hebben op DRM

Flash Player 10.1 of hoger, Adobe AIR 1.0 of hoger

De klasse NetStream vormt een streamingverbinding in één richting tussen enerzijds Flash Player of een AIR-toepassing en anderzijds de Flash Media Server of het lokale bestandssysteem. (De klasse NetStream ondersteunt ook progressief downloaden.) Een NetStream-object is een kanaal binnen een NetConnection-object. De NetStream-klasse verzendt vier DRM-gerelateerde gebeurtenissen:

Gebeurtenis	**Beschrijving**
drmAuthenticate (Alleen AIR)	Gedefinieerd in de DRMAuthenticateEvent-klasse. Deze gebeurtenis wordt verzonden wanneer een NetStream-object beveiligde inhoud wil afspelen waarvoor gebruikersverificatie is vereist voordat de inhoud kan worden afgespeeld. De eigenschappen van deze gebeurtenissen zijn header, usernamePrompt, passwordPrompt en urlPrompt, waarmee de gebruikersgegevens kunnen worden verkregen en ingesteld. Deze gebeurtenis wordt herhaald tot het NetStream-object geldige gebruikersgegevens ontvangt.
drmError	Gedefinieerd in de DRMErrorEvent-klasse en verzonden wanneer een NetStream-object beveiligde inhoud wil afspelen en een DRM-gerelateerde fout optreedt. Voorbeeld: er wordt een gebeurtenisobject voor de DRM-fout verzonden als de gebruikersautorisatie mislukt. Deze fout kan zich voordoen wanneer de gebruiker geen rechten heeft gekocht om de inhoud weer te geven. Deze fout kan zich ook voordoen als de inhoudsprovider geen ondersteuning biedt voor de weergavetoepassing.
drmStatus	Gedefinieerd in de DRMStatusEvent-klasse. Deze gebeurtenis wordt verzonden wanneer de beveiligde inhoud wordt afgespeeld (wanneer de gebruiker is geverifieerd en geautoriseerd om de inhoud af te spelen). Het DRMStatusEvent-object bevat informatie die is gerelateerd aan de voucher. Deze informatie geeft onder meer aan of de inhoud offline toegankelijk mag worden gemaakt, of wanneer de voucher vervalt en de inhoud niet langer mag worden bekeken.
status	Gedefinieerd in events.StatusEvent en alleen verzonden wanneer de toepassing probeert om beveiligde inhoud af te spelen door de methode NetStream.play() aan te roepen. De waarde van de statuscode-eigenschap is "DRM.encryptedFLV".

De NetStream-klasse bevat onder andere de volgende DRM-specifieke methoden, alleen voor gebruik in AIR:

Methode	Beschrijving
resetDRMVouchers()	Verwijdert alle lokaal in de cache geplaatste DRM-bongegevens. De toepassing moet de bonnen opnieuw downloaden, anders kan de gebruiker de gecodeerde inhoud niet openen. De volgende code verwijdert bijvoorbeeld alle vouchers uit de cache: NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	Geeft een reeks verificatiegegevens (zoals gebruikersnaam, wachtwoord en verificatietype) voor verificatie door aan het NetStream-object. Geldige verificatietypen zijn `''drm''` en `''proxy''`. Bij het verificatietype `"drm"` worden de gegevens geverifieerd met behulp van Adobe Access. Bij het verificatietype `"proxy"` worden de gegevens geverifieerd met behulp van de proxyserver. Ze moeten overeenkomen met de gegevens die door de proxyserver worden vereist. Een onderneming kan bijvoorbeeld eisen dat de toepassing moet worden geverifieerd met behulp van een proxyserver, voordat de gebruiker toegang tot internet krijgt. Dit type verificatie is mogelijk met de optie "proxy". Tenzij anonieme verificatie wordt gebruikt, moet de gebruiker na de proxyverificatie nog steeds door Adobe Access worden geverifieerd om de voucher te verkrijgen en de inhoud af te spelen. U kunt `setDRMAuthenticationCredentials()` nogmaals gebruiken (nu met de optie `"drm"`) om verificatie uit te voeren met behulp van Adobe Access.
preloadEmbeddedMetadata()	Parseert een lokaal mediabestand op ingesloten metagegevens. Wanneer metagegevens worden gevonden die betrekking hebben op DRM, roept AIR de callback-functie `onDRMContentData()` aan.

Verder roept een NetStream-object in AIR de callbackfuncties `onDRMContentData()` en `onPlayStatus()` aan als resultaat van een aanroep naar de methode `preloadEmbeddedMetaData()`. De functie `onDRMContentData()` wordt aangeroepen wanneer DRM-metagegevens worden aangetroffen in een mediabestand. De functie `onPlayStatus()` wordt aangeroepen wanneer het bestand is geparseerd. De functies `onDRMContentData()` en `onPlayStatus()` moeten worden gedefinieerd op het `client`-object dat is toegewezen aan de NetStream-instantie. Als u hetzelfde NetStream-object gebruikt om metagegevens op voorhand te laden en om inhoud af te spelen, moet u met het afspelen wachten op de aanroep van `onPlayStatus()` die door `preloadEmbeddedMetaData()` wordt gegenereerd.

In de volgende code van AIR zijn de gebruikersnaam ("administrator"), het wachtwoord ("password") en het verificatietype "drm" ingesteld om de gebruiker te verifiëren. De methode setDRMAuthenticationCredentials() moet verificatiegegevens bieden die overeenkomen met de verificatiegegevens die bekend zijn bij en worden geaccepteerd door de inhoudsprovider. Het gaat hierbij om dezelfde verificatiegegevens waarmee de gebruiker de inhoud kan bekijken. De code om de video af te spelen en het controleren of een verbinding met de videostroom tot stand is gebracht, is hier niet opgenomen.

```
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");
}
```

De klasse DRMStatusEvent gebruiken

Flash Player 10.1, Adobe AIR 1.0 en hoger

Een NetStream-object verzendt een DRMStatusEvent-object wanneer de door Adobe Access-beveiligde inhoud succesvol wordt afgespeeld. (Succesvol houdt in dat de licentie is geverifieerd en dat de gebruiker is geverifieerd en geautoriseerd om de inhoud te bekijken). Het DRMStatusEvent-object wordt ook voor anonieme gebruikers verzonden als deze toegang mogen hebben. Met de licentie wordt gecontroleerd of anonieme gebruikers, die geen verificatie nodig hebben, toegang krijgen om de inhoud af te spelen. Aan anonieme gebruikers kan deze toegang om verschillende redenen worden ontzegd. Voorbeeld: een anonieme gebruiker krijgt geen toegang tot de inhoud omdat de licentie verstreken is.

Het DRMStatusEvent-object bevat informatie die is gerelateerd aan de licentie. Deze informatie geeft onder meer aan of de licentie offline toegankelijk mag worden gemaakt of wanneer de voucher vervalt en de inhoud niet langer mag worden bekeken. De toepassing kan deze gegevens gebruiken om de beleidsstatus en de toestemmingen van de gebruiker door te geven.

DRMStatusEvent-eigenschappen

Flash Player 10.1, Adobe AIR 1.0 en hoger

De klasse DRMStatusEvent bevat de volgende eigenschappen. Sommige eigenschappen zijn pas beschikbaar in hogere versies van AIR dan versie 1.0. Zie de *ActionScript 3.0 Reference* voor de volledige informatie over versies.

Voor eigenschappen die niet worden ondersteund in Flash Player 10.1 biedt de DRMVoucher-klasse vergelijkbare eigenschappen voor Flash Player.

Eigenschap	**Beschrijving**
contentData	Een DRMContentData-object dat de DRM-metagegevens bevat die in de inhoud zijn ingesloten.
detail (alleen AIR)	Een tekenreeks die de context van de statusgebeurtenis verklaart. In DRM 1.0 is DRM.voucherObtained de enige geldige waarde.
isAnonymous (alleen AIR)	Geeft aan of de inhoud die met Adobe Access is beveiligd, toegankelijk is zonder dat de verificatiegegevens van de gebruiker hoeven te worden opgevraagd (true) of dat deze wel moeten worden opgevraagd (false). De waarde "false" betekent dat de gebruiker een gebruikersnaam en wachtwoord moet opgeven die overeenkomen met de gebruikersnaam die en het wachtwoord dat bij de inhoudsprovider bekend zijn en door deze worden verwacht.
isAvailableOffline (alleen AIR)	Geeft aan of de inhoud die met adobe Acces is beveiligd, offline (true) of niet (false) toegankelijk mag worden gemaakt. Om digitaal beveiligde inhoud offline toegankelijk te kunnen maken, moet de bijbehorende bon in een cache worden geplaatst op het lokale toestel van de gebruiker.
isLocal	Geeft aan of de voucher die nodig is om de inhoud af te spelen, in de lokale cache is opgeslagen.
offlineLeasePeriod (alleen AIR)	Het resterende aantal dagen dat de inhoud offline mag worden bekeken.
policies (alleen AIR)	Een aangepast object dat aangepaste DRM-eigenschappen kan bevatten.
voucher	De DRMVoucher.
voucherEndDate (alleen AIR)	De absolute datum waarop de bon vervalt en de inhoud dus niet langer mag worden bekeken.

DRMStatusEvent-handlers maken

Flash Player 10.1, Adobe AIR 1.0 en hoger

In het volgende voorbeeld ziet u hoe u een gebeurtenishandler maakt die de statusinformatie van de DRM-inhoud doorstuurt naar het NetStream-object dat de gebeurtenis heeft gestart. Voeg deze gebeurtenishandler toe aan een NetStream-object dat naar de beveiligde inhoud wijst.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

De klasse DRMAuthenticateEvent gebruiken

Adobe AIR 1.0 of hoger

Het DRMAuthenticateEvent-object wordt verzonden wanneer een NetStream-object beveiligde inhoud wil afspelen waarvoor gebruikersverificatie is vereist voordat de inhoud kan worden afgespeeld.

De DRMAuthenticateEvent-handler verzamelt de vereiste gegevens (gebruikersnaam, wachtwoord en type) en geeft deze waarden voor verificatie door aan de methode `NetStream.setDRMAuthenticationCredentials()`. Elke AIR-toepassing moet over een mechanisme beschikken om de gebruikersgegevens te verkrijgen. De toepassing kan bijvoorbeeld een eenvoudige gebruikersinterface voor het invoeren van de gebruikersnaam en het wachtwoord weergeven. De toepassing kan ook een mechanisme bieden voor het omgaan met en beperken van herhaalde verificatiepogingen.

DRMAuthenticateEvent-eigenschappen

Adobe AIR 1.0 of hoger

De klasse DRMAuthenticateEvent bevat de volgende eigenschappen:

Eigenschap	Beschrijving
authenticationType	Geeft aan of de geleverde gegevens bedoeld zijn voor verificatie met behulp van Adobe Access ("drm") of met behulp van een proxyserver ("proxy"). De optie "proxy" geeft aan dat de toepassing indien nodig moet worden geverifieerd met behulp van een proxyserver, voordat de gebruiker toegang krijgt tot internet. Tenzij anonieme verificatie wordt gebruikt, moet de gebruiker na de proxyverificatie nog steeds door Adobe Access worden geverifieerd om de voucher te verkrijgen en de inhoud af te spelen. U kunt setDRMAuthenticationCredentials() nogmaals gebruiken (nu met de optie "drm") om verificatie uit te voeren met behulp van Adobe Access.
header	De bestandsheader van de gecodeerde inhoud, geleverd door de server. Bevat informatie over de context van de gecodeerde inhoud. Deze headertekenreeks kan worden doorgegeven aan de Flash-toepassing, zodat deze een dialoogvenster voor het opgeven van een wachtwoord en gebruikersnaam kan samenstellen. De headertekenreeks kan worden gebruikt voor de instructies in het dialoogvenster. De header zou bijvoorbeeld "Voer uw gebruikersnaam en wachtwoord in" kunnen zijn.
netstream	Het NetStream-object dat deze gebeurtenis heeft geactiveerd.
passwordPrompt	Een prompt voor wachtwoordgegevens, geleverd door de server. De tekenreeks kan een instructie bevatten voor het vereiste type wachtwoord.
urlPrompt	Een prompt voor een URL-tekenreeks, geleverd door de server. De tekenreeks kan de locatie aangeven waarnaar de gebruikersnaam en het wachtwoord worden verzonden.
usernamePrompt	Een prompt voor gebruikersnaamgegevens, geleverd door de server. De tekenreeks kan een instructie bevatten voor het vereiste type gebruikersnaam. Een inhoudsprovider kan bijvoorbeeld een e-mailadres als gebruikersnaam vereisen.

De eerder vermelde tekenreeksen worden alleen verschaft door de FMRMS-server. De Adobe Access-server gebruikt deze tekenreeksen niet.

DRMAuthenticateEvent-handlers maken

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u een gebeurtenishandler maakt die een reeks hard-gecodeerde autorisatiegegevens doorgeeft aan het NetStream-object dat de gebeurtenis heeft gestart. (De code om de video af te spelen en het controleren of een verbinding met de videostroom tot stand is gebracht, is hier niet opgenomen.)

```
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");
}
```

Interfaces maken om gebruikersgegevens op te vragen

Adobe AIR 1.0 of hoger

Als gebruikersverificatie vereist is voor beveiligde inhoud, moet de AIR-toepassing de verificatiegegevens van de gebruiker doorgaans via een gebruikersinterface opvragen.

Hierna volgt een Flex-voorbeeld van een eenvoudige gebruikersinterface om de gebruikersgegevens op te vragen. Deze interface bestaat uit een deelvensterobject met twee TextInput-objecten: één voor de gebruikersnaam en één voor het wachtwoord. Het deelvenster bevat ook een knop waarmee de methode `credentials()` wordt gestart.

```
<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>
```

De methode `credentials()` is een door de gebruiker gedefinieerde methode waarmee de gebruikersnaam en het wachtwoord worden doorgegeven aan de methode `setDRMAuthenticationCredentials()`. Nadat de waarden zijn doorgestuurd, voert de methode `credentials()` een reset uit van de waarden van de TextInput-objecten.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

U kunt een dergelijke eenvoudige interface implementeren door het deelvenster als onderdeel van een nieuwe status op te nemen. De nieuwe status komt voort uit de basisstatus wanneer het object DRMAuthenticateEvent wordt gegenereerd. In het volgende voorbeeld ziet u een VideoDisplay-object met een bronkenmerk dat naar een beveiligd FLV-bestand wijst. In dit geval wordt de methode `credentials()` gewijzigd, zodat ook de toepassing naar de basisstatus wordt geretourneerd. Dit wordt gedaan door deze methode nadat de verificatiegegevens van de gebruiker zijn doorgegeven en de TextInput-objectwaarden opnieuw zijn ingesteld.

```
<?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>
```

De klasse DRMErrorEvent gebruiken

Flash Player 10.1 of hoger, Adobe AIR 1.0 of hoger

Adobe Flash Player en Adobe AIR verzenden een DRMErrorEvent-object wanneer een NetStream-object beveiligde inhoud wil afspelen en er een DRM-gerelateerde fout optreedt. Als verificatiegegevens van de gebruiker ongeldig zijn in een AIR-toepassing, wordt het DRMAuthenticateEvent-object herhaaldelijk verzonden totdat de gebruiker geldige verificatiegegevens invoert of geen nieuwe poging meer wordt toegestaan door de toepassing. De toepassing is verantwoordelijk voor het luisteren naar eventuele andere DRM-foutgebeurtenissen om de aan DRM gerelateerde fouten te detecteren, te identificeren en af te handelen.

Zelfs als een gebruiker geldige gebruikersgegevens heeft, kan er op grond van de voorwaarden van de voucher van de inhoud worden voorkomen dat een gebruiker de gecodeerde inhoud kan bekijken. Een gebruiker kan bijvoorbeeld de toegang worden geweigerd als deze inhoud probeert te bekijken in een niet-geautoriseerde toepassing. Een niet-geautoriseerde toepassing is een toepassing die de uitgever van de gecodeerde inhoud niet heeft gevalideerd. In dat geval wordt een DRMErrorEvent-object verzonden.

De foutgebeurtenissen kunnen ook worden gegenereerd als de inhoud beschadigd is of als de versie van de toepassing niet overeenkomt met de versie die door de voucher wordt aangegeven. De toepassing moet over een mechanisme beschikken om fouten af te handelen.

DRMErrorEvent-eigenschappen

Flash Player 10.1 of hoger, Adobe AIR 1.0 of hoger

Zie Runtimefoutcodes in de ActionScript 3.0 Reference voor de volledige lijst met fouten. De DRM-gerelateerde fouten beginnen bij fout 3300.

DRMErrorEvent-handlers maken

Flash Player 10.1 of hoger, Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u een gebeurtenishandler maakt voor het NetStream-object dat de gebeurtenis heeft gestart. Deze gebeurtenishandler wordt aangeroepen als de NetStream een fout aantreft wanneer wordt geprobeerd om beveiligde inhoud af te spelen. Als een toepassing een fout aantreft, wordt er doorgaans een aantal opschoningstaken uitgevoerd. Vervolgens stelt de toepassing de gebruiker op de hoogte van de fout en worden er opties geboden voor het oplossen van het probleem.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

De klasse DRMManager gebruiken

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

Gebruik de klasse DRMManager voor het beheer van vouchers en sessies voor de mediarechtenserver in toepassingen.

Voucherbeheer (alleen AIR)

Wanneer een gebruiker beveiligde inhoud afspeelt, wordt de licentie voor het bekijken van de inhoud opgehaald door het runtimeprogramma en in het cachegeheugen opgeslagen. Als de toepassing het bestand lokaal opslaat en offline afspelen door de licentie wordt toegestaan, kan de gebruiker de inhoud in de AIR-toepassing weergeven. Offline afspelen is op deze manier zelfs mogelijk als er geen verbinding met de mediarechtenserver is. Met de DRMManager en de NetStream `preloadEmbeddedMetadata()`-methode kunt u de voucher vooraf opslaan in het cachegeheugen. De toepassing hoeft de benodigde licentie niet op te halen om de inhoud te bekijken. Uw toepassing zou bijvoorbeeld het mediabestand kunnen downloaden en dan de bon ophalen terwijl de gebruiker nog online is.

Om een bon vooraf te laden, gebruikt u de methode `preloadEmbeddedMetadata()` van NetStream om een DRMContentData-object te verkrijgen. Het DRMContentData-object bevat de URL en het domein van de mediarechtenserver die de licentie kan verstrekken. Verder wordt in dit object aangegeven of verificatie van de gebruiker vereist is. Met deze informatie kunt u de methode `loadVoucher()` van DRMManager aanroepen om de bon te verkrijgen en in de cache op te slaan. De workflow voor het vooraf laden van vouchers wordt gedetailleerd beschreven in “Vouchers voor offline afspelen vooraf laden” op pagina 561.

Sessiebeheer

U kunt de DRMManager ook gebruiken om de gebruiker te verifiëren bij een mediarechtenserver en om blijvende sessies te beheren.

Roep de methode `authenticate()` van DRMManager aan om een sessie met de mediarechtenserver tot stand te brengen. Wanneer de verificatie voltooid is, verzendt de DRMManager een DRMAuthenticationCompleteEvent-object. Dit object bevat een sessietoken. U kunt dit token opslaan om toekomstige sessies mee tot stand te brengen zodat de gebruiker niet zijn accountdetails hoeft op te geven. Geef het token door aan de methode `setAuthenticationToken()` om een nieuwe geverifieerde sessie tot stand te brengen. (De instellingen van de server die het token heeft gegenereerd, bepalen wanneer het token vervalt en andere kenmerken. AIR-toepassingscode behoort de tokengegevensstructuur niet te interpreteren, omdat de structuur in toekomstige AIR-updates gewijzigd kan worden.)

Verificatietokens kunnen worden overgedragen naar andere computers. Om tokens te beschermen, kunt u ze opslaan in de door AIR gecodeerde lokale opslag. Zie “Lokale gegevens gecodeerd opslaan” op pagina 750 voor meer informatie.

DRMStatus-gebeurtenissen

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

De DRMManager verzendt een DRMStatusEvent-object nadat een aanroep naar de methode `loadVoucher()` met succes wordt voltooid.

Als een voucher is verkregen, heeft de eigenschap `detail` (alleen AIR) van het gebeurtenisobject de waarde: "DRM.voucherObtained", en bevat de eigenschap `voucher` het DRMVoucher-object.

Als geen voucher is verkregen, heeft de eigenschap `detail` (alleen AIR) nog steeds de volgende waarde: "DRM.voucherObtained"; de eigenschap `voucher` is dan echter `null`. Een voucher kan niet worden verkregen als u de LoadVoucherSetting van *localOnly* gebruikt en er geen voucher lokaal in de cache is opgeslagen.

Als de aanroep naar `loadVoucher()` niet met succes is voltooid, bijvoorbeeld vanwege een verificatie- of communicatiefout, verzendt de DRMManager in plaats daarvan een DRMErrorEvent- of DRMAuthenticationErrorEvent-object.

DRMAuthenticationComplete-gebeurtenissen

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

De DRMManager verzendt een DRMAuthenticationCompleteEvent-object wanneer een gebruiker met succes is geverifieerd middels een aanroep van de methode `authenticate()`.

Het DRMAuthenticationCompleteEvent-object bevat een herbruikbaar token dat kan worden gebruikt om de verificatie van de gebruiker blijvend te maken over meerdere toepassingssessies. Geef deze token door aan de DRMManager-methode `setAuthenticationToken()` als u een sessie opnieuw tot stand wilt brengen. (De maker van het token stelt de kenmerken hiervan in. Adobe heeft geen API om tokenkenmerken te controleren.)

DRMAuthenticationError-gebeurtenissen

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

De DRMManager verzendt een DRMAuthenticationErrorEvent-object wanneer een gebruiker niet kan worden geverifieerd via een aanroep naar de methode `authenticate()` of `setAuthenticationToken()`.

De klasse DRMContentData gebruiken

Flash Player 10.1 of hoger, Adobe AIR 1.5 of hoger

Het DRMContentData-object bevat de metagegevenskenmerken van inhoud die is beveiligd door adobe Access. De DRMContentData-eigenschappen bevatten de informatie die nodig is om een licentievoucher voor het weergeven van de inhoud te verkrijgen. Met de DRMContentData-klasse kunt u het metagegevensbestand verkrijgen dat is gekoppeld aan de inhoud, zoals beschreven in de "Gedetailleerde API-workflow" op pagina 558.

Zie de DRMContentData-klasse in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

Een update voor Flash Player uitvoeren voor ondersteuning van Adobe Access

Flash Player 10.1 of hoger

Belangrijk: Flash Player 11.5 en hoger integreert de Adobe Access-module, zodat een update (het aanroepen van `SystemUpdater.update(SystemUpdaterType.DRM)`) onnodig is. Dit omvat de volgende browsers en platforms:

- Flash Player 11.5 ActiveX-besturingselement voor alle platforms behalve Internet Explorer op Windows 8
- Flash Player 11.5-insteekmodule voor alle browsers
- Adobe AIR (voor desktop en mobiel)

Dit betekent dat de update nog *steeds vereist* is in de volgende gevallen:

- Internet Explorer op Windows 8
- Flash Player 11.4 en lager, behalve op Google Chrome 22 en hoger (alle platforms) of 21 en hoger (Windows)

Opmerking: *U kunt* `SystemUpdater.update(SystemUpdaterType.DRM)` *nog steeds veilig aanroepen op een systeem met Flash Player 11.5 of hoger, maar er wordt niet gedownload.*

Voor ondersteuning van Adobe Access vereist Flash Player de Adobe Access-module. Wanneer Flash Player probeert beveiligde inhoud af te spelen, geeft het runtimeprogramma aan of de module of een nieuwe versie van Flash Player moet worden gedownload. Zo hebben SWF-ontwikkelaars de keuze om Flash Player al dan niet bij te werken, geheel naar wens.

In de meeste gevallen werken SWF-ontwikkelaars voor het afspelen van beveiligde inhoud de vereiste Adobe Access-module of Flash Player bij. Als u een update wilt uitvoeren, gebruikt u de SystemUpdater-API om de laatste versie van de Adobe Access-module of van Flash Player op te halen.

Met de SystemUpdater-API kunt u slechts een update tegelijkertijd uitvoeren. De foutcode 2202 geeft aan dat er al een update wordt uitgevoerd in de huidige runtime-instantie of in een andere instantie. Als bijvoorbeeld een Flash Player-instantie in Internet Explorer wordt bijgewerkt, kan een Flash Player-instantie die tegelijkertijd in Firefox wordt uitgevoerd niet worden bijgewerkt.

De SystemUpdater-API wordt alleen ondersteund op bureaubladplatforms.

Opmerking: *Voor eerdere versies van Flash Player dan versie 10.1 moet u het updatemechanisme gebruiken dat in de eerdere versies wordt ondersteund (handmatig downloaden en installeren vanaf www.adobe.com of ExpressInstall). Het installatieprogramma van AIR verwerkt de benodigde updates voor Adobe Access en biedt geen ondersteuning voor de SystemUpdater-API.*

Luisteren naar een updategebeurtenis

Flash Player 10.1 of hoger

Wanneer een update van de Adobe Access-module is vereist, verzendt het NetStream-object een NetStatusEvent met de codewaarde `DRM.UpdateNeeded`. Deze waarde geeft aan dat het NetStream-object niet de beveiligde stream kan afspelen met een van de Adobe Access-modules die momenteel zijn geïnstalleerd. Luister naar deze gebeurtenis en roep de volgende code aan:

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

Met deze code wordt de Adobe Access-module bijgewerkt die in Flash Player is geïnstalleerd. Er is geen gebruikerstoestemming nodig voor deze module-update.

Als de Adobe Access-module niet wordt gevonden, wordt een foutmelding gegenereerd. Zie stap 3 van de “Gedetailleerde API-workflow” op pagina 558.

Opmerking: *Als play() wordt aangeroepen op een gecodeerde stream in Flash Player-versies voor versie 10.1, wordt een NetStatusEvent met de codewaarde NetStream.Play.StreamNotFound verzonden. Bij eerdere versies moet u het updatemechanisme gebruiken dat wordt ondersteund voor de desbetreffende versies (handmatig downloaden en installeren van www.adobe.com of ExpressInstall).*

Wanneer een Flash Player-update is vereist, verzendt het SystemUpdater-object een StatusEvent met de codewaarde `DRM.UpdateNeededButIncompatible`. Voor het uitvoeren van een Flash Player-update is gebruikerstoestemming vereist. Zorg ervoor dat uw toepassing beschikt over een interface waarmee de gebruiker zijn toestemming kan geven aan een update van Flash Player en waarmee deze update kan worden geïnitieerd. Luister naar de StatusEvent-gebeurtenis en roep de volgende code aan:

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

Met deze code wordt de update voor Flash Player geïnitieerd.

Aanvullende gebeurtenissen voor de SystemUpdater-klasse worden beschreven in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform.

Nadat Flash Player is bijgewerkt, wordt de gebruiker verwezen naar de pagina waar de update is gestart. De Adobe Access-module is gedownload en de stream kan worden afgespeeld.

Offlinelicenties

Flash Player 11 of hoger, Adobe AIR 3.0 of hoger

Het is ook mogelijk licenties offline te verkrijgen (zonder contact op te nemen met een Adobe Access-licentieserver) door de voucher (licentie) via de methode `storeVoucher` op schijf en in het geheugen op te slaan.

Als u gecodeerde video's wilt afspelen in Flash Player en AIR, dient de respectievelijke runtime de DRM-voucher voor die video te verkrijgen. De DRM-voucher bevat de coderingssleutel van de video en wordt gegenereerd door de Adobe Access-licentieserver die de klant heeft geïmplementeerd.

De Flash Player/AIR-runtime verkrijgt deze voucher meestal door een voucherverzoek te verzenden naar de Adobe Access-licentieserver die wordt vermeld in de DRM-metagegevens (`DRMContentData`-klasse) van de video. De Flash/AIR-toepassing kan deze licentieaanvraag activeren door de `DRMManager.loadVoucher()`-methode aan te roepen. Het is ook mogelijk dat de Flash Player/AIR-runtime automatisch een licentie aanvraagt wanneer wordt begonnen met afspelen van de gecodeerde video als er geen licentie voor de inhoud in het geheugen of op schijf staat. In beide gevallen worden de prestaties van de Flash/AIR-toepassing beïnvloed door de communicatie met de Adobe Access-licentieserver.

`DRMManager.storeVoucher()` stelt de Flash/AIR-toepassing in staat offline verkregen DRM-vouchers te verzenden naar de Flash Player/AIR-runtime. De runtime kan het licentieaanvraagproces dan overslaan en de doorgestuurde vouchers gebruiken om gecodeerde video's af te spelen. De DRM-voucher moet echter nog steeds gegenereerd worden door de Adobe Access-licentieserver voordat deze offline kan worden verkregen. Het is echter mogelijk de vouchers op een willekeurige HTTP-server te hosten in plaats van op een publiek toegankelijke Adobe Access-licentieserver.

`DRMManager.storeVoucher()` wordt ook gebruikt voor ondersteuning van het uitwisselen van DRM-vouchers tussen meerdere apparaten. In Adobe Access 3.0 wordt deze functie "Domeinondersteuning" genoemd. Als uw implementatie ondersteuning biedt voor deze manier van gebruik, kunt u meerdere apparaten registreren voor een apparaatgroep met gebruik van de methode `DRMManager.addToDeviceGroup()`. Als een van de apparaten over een geldige domeingebonden voucher voor bepaalde inhoud beschikt, kan de AIR-toepassing de geserialiseerde DRM-vouchers vervolgens extraheren met de methode `DRMVoucher.toByteArray()`. Op uw andere apparaten kunt u de vouchers importeren met gebruik van de methode `DRMManager.storeVoucher()`.

Apparaten registreren

De DRM-vouchers zijn gebonden aan het apparaat van de eindgebruiker. Daarom hebben Flash/AIR-toepassingen een unieke id voor de computer van de gebruiker nodig, anders kunnen ze niet naar het juiste geserialiseerde DRM-voucherobject verwijzen. Het volgende scenario beschrijft een registratieprocedure voor een apparaat.

Er wordt van uitgegaan dat u de volgende taken hebt uitgevoerd:

- U hebt de SDK van de Adobe Access-server ingesteld.
- U hebt een HTTP-server ingesteld voor het verkrijgen van vooraf gegenereerde licenties.
- U hebt een Flash-toepassing gemaakt voor weergave van de beveiligde inhoud.

Bij de registratiefase van het apparaat komen de volgende handelingen kijken:

1 De Flash-toepassing creëert een willekeurig gegenereerde id.

2 De Flash-toepassing roept de methode `DRMManager.authenticate()` aan. De toepassing dient de willekeurig gegenereerde id in het verificatieverzoek op te nemen. Neem de id bijvoorbeeld op in het veld voor de gebruikersnaam.

3 De in Stap 2 vermelde handeling leidt ertoe dat Adobe Access een verificatieverzoek naar de server van de klant stuurt. Het apparaatcertificaat is in dit verzoek opgenomen.

   a De server extraheert het apparaatcertificaat en de gegenereerde id uit het verzoek en slaat deze op.

   b Het subsysteem van de klant genereert vooraf licenties voor dit apparaatcertificaat, slaat deze op en verleent toegang tot deze licenties op basis van associatie met de gegenereerde id.

4 De server reageert met een "succes"-bericht op het verzoek.

5 De Flash-toepassing slaat de gegenereerde id lokaal op in een LSO (Local Shared Object).

Na de apparaatregistratie gebruikt de Flash-toepassing de gegenereerde id op dezelfde wijze als de apparaat-id in het vorige schema zou zijn gebruikt:

1 De Flash-toepassing probeert de gegenereerde id te vinden in LSO.

2 Wanneer de gegenereerde id wordt gevonden, gebruikt de Flash-toepassing de gegenereerde id terwijl de vooraf gegenereerde licenties worden gedownload. De Flash-toepassing verzendt de licenties naar de Adobe Access-client, zodat deze kunnen worden gebruikt met de methode `DRMManager.storeVoucher()`.

3 Wanneer de gegenereerde id niet wordt gevonden, doorloopt de Flash-toepassing de procedure voor apparaatregistratie.

Fabriekinstellingen herstellen

Wanneer de gebruiker van het apparaat de optie voor het herstellen van de fabrieksinstellingen aanroept, wordt het apparaatcertificaat gewist. Als u de beschermde inhoud wilt blijven afspelen, dient de Flash-toepassing de registratieprocedure voor het apparaat nogmaals te doorlopen. Als de Flash-toepassing een verouderde, vooraf gegenereerde licentie verzendt, zal de Adobe Access-client deze afwijzen, aangezien de licentie voor een oudere apparaat-id was gecodeerd.

Domeinondersteuning

Flash Player 11 of hoger, Adobe AIR 3.0 of hoger

Als de metagegevens voor de inhoud aangeven dat domeinregistratie is vereist, kan de AIR-toepassing een API aanroepen om zich te kunnen aansluiten bij een apparaatgroep. Deze handeling leidt ertoe dat er een domeinregistratieverzoek naar de domeinserver wordt verzonden. Een eenmaal aan een apparaatgroep verleende licentie kan worden geëxporteerd en gedeeld met andere apparaten die zich bij de apparaatgroep hebben aangesloten.

De apparaatgroepinformatie wordt vervolgens gebruikt in het `VoucherAccessInfo`-object van `DRMContentData` dat vervolgens weer wordt gebruikt voor het weergeven van de informatie die nodig is om een voucher te kunnen ophalen en gebruiken.

Gecodeerde inhoud afspelen met gebruik van domeinondersteuning

Voer de volgende stappen uit om gecodeerde inhoud te kunnen afspelen met gebruik van Adobe Access:

1. Controleer aan de hand van `VoucherAccessInfo.deviceGroup` of apparaatgroepregistratie is vereist.
2. Als dat het geval is:
   a. Gebruik de eigenschap `DeviceGroupInfo.authenticationMethod` om te zien of verificatie is vereist.
   b. Indien verificatie is vereist, verifieert u de gebruiker door EEN van de volgende stappen uit te voeren:
      - Haal de gebruikersnaam en het wachtwoord van de gebruiker op. Roep `DRMManager.authenticate (deviceGroup.serverURL, deviceGroup.domain, gebruikersnaam, wachtwoord)` aan.
      - Haal een in de cache opgeslagen of vooraf gegenereerde verificatietoken op en roep `DRMManager.setAuthenticationToken()` aan.
   c. Roep `DRMManager.addToDeviceGroup()` aan.
3. Haal de voucher voor de inhoud op door een van de volgende taken uit te voeren:
   a. Gebruik de `DRMManager.loadVoucher()`-methode.
   b. Haal de voucher op van een ander apparaat dat in dezelfde apparaatgroep is geregistreerd. Geef de voucher door aan `DRMManager` aan de hand van de methode `DRMManager.storeVoucher()`.
4. Speel de gecodeerde inhoud af met gebruik van de `NetStream.play()` -methode.

Als u de licentie voor de inhoud wilt exporteren, kan elk apparaat u de onbewerkte bytes van de licentie verschaffen via de `DRMVoucher.toByteArray()`-methode nadat u de licentie van de Adobe Access-licentieserver hebt verkregen. Inhoudsproviders beperken doorgaans het aantal apparaten in een apparaatgroep. Wanneer deze limiet is bereikt, dient u wellicht de methode `DRMManager.removeFromDeviceGroup()` aan te roepen voor een apparaat dat u niet gebruikt voordat u het huidige apparaat kunt registreren.

Voorvertoning van licentie

De Flash-toepassing kan een verzoek voor een voorvertoning van de licentie verzenden. Dit betekent dat de toepassing een voorvertoning kan uitvoeren voordat de gebruiker wordt gevraagd de inhoud aan te schaffen. Op deze manier kan worden bepaald of de pc van de gebruiker daadwerkelijk aan alle afspeelcriteria voldoet. Een licentievoorvertoning betekent dat de klant een voorvertoning van de licentie kan zien (om te zien welke rechten een licentie geeft), niet dat de klant een voorvertoning van de inhoud kan zien (een gedeelte van de inhoud weergeven voordat de klant besluit of hij een licentie koopt). Hier volgen enkele parameters die uniek zijn voor elke computer: beschikbare uitvoer en de beveiligingsstatus hiervan, de beschikbare runtime/DRM-versie en het beveiligingsniveau van de DRM-client. De voorvertoningsmodus voor licenties maakt het de runtime/DRM-client mogelijk de bedrijfslogica van de licentieserver te testen en om informatie terug te sturen naar de gebruiker, zodat deze een goed geïnformeerde beslissing kan nemen. De klant kan zo zien hoe een geldige licentie eruitziet, maar ontvangt de sleutel voor het decoderen van de inhoud nog niet. Ondersteuning voor het voorvertonen van licenties is optioneel en alleen noodzakelijk als u een aangepaste toepassing implementeert die gebruikmaakt van deze functie.

Inhoud leveren

Adobe Access is niet op de hoogte van de methode waarop de inhoud wordt geleverd, aangezien de Flash Player de netwerklaag onttrekt en de beveiligde inhoud gewoon aan het Adobe Access-subsysteem verschaft. Inhoud kan dus worden geleverd via HTTP, HTTP Dynamic Streaming, RTMP, of RTMPE.

Er kunnen echter enkele problemen optreden vanwege de noodzakelijke metagegevens van de beveiligde inhoud (meestal in de vorm van een bestand .metadata) voordat Adobe Access een licentie voor het decoderen van de inhoud kan aanschaffen. Specifiek kunnen met het RTMP/RTMPE-protocol alleen FLV- en F4V-gegevens aan de client worden geleverd via de FMS (Flash Media Server). Daarom dient de client de metagegevensblob op andere manieren op te halen. U kunt dit probleem oplossen door de metagegevens te hosten op een HTTP-webserver en de videospeler van de client te implementeren om de desbetreffende metagegevens op te halen, afhankelijk van de inhoud die wordt afgespeeld.

```
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

OSMF (Open Source Media Framework) is een op ActionScript gebaseerd raamwerk dat u de flexibiliteit en controle geeft om veelzijdige mediaervaringen te maken. Ga voor meer informatie over OSMF naar de website voor OSMF-ontwikkelaars.

Workflow voor het afspelen van beveiligde inhoud

1 Maak een `MediaPlayer`-instantie.

```
player = new MediaPlayer();
```

2 Registreer de `MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE`-gebeurtenis bij de speler. Deze gebeurtenis wordt verzonden als de inhoud via DRM is beveiligd.

```
player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE,
onDRMCapabilityChange);
```

3 Haal de `DRMTrait`-instantie op in een gebeurtenishandler. `DRMTrait` is de interface die u gebruikt voor het aanroepen van aan DRM gerelateerde methoden, zoals `authenticate()`. Tijdens het laden van door DRM beveiligde inhoud, voert OSMF de handelingen voor DRM-validatie uit en verzendt het statusgebeurtenissen. Voeg een `DRMEvent.DRM_STATE_CHANGE`-gebeurtenishandler toe aan de `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 Handel de DRM-gebeurtenissen af in de `onDRMStateChange()`-methode.

```
        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;
    }
}
```

Hoofdstuk 28: PDF-inhoud toevoegen in AIR

Adobe AIR 1.0 of hoger

Actieve toepassingen in Adobe® AIR® kunnen niet alleen SWF- en HTML-inhoud, maar ook PDF-inhoud renderen. AIR-toepassingen renderen PDF-inhoud met behulp van de klasse HTMLLoader, de WebKit-engine en de Adobe® Reader® plug-in voor de browser. In een AIR-toepassing kan PDF-inhoud gelden voor de volledige hoogte en breedte van uw toepassing of slechts voor een deel van de interface. Met de Adobe Reader-plug-in voor de browser kunt instellen hoe PDF-bestanden in een AIR-toepassing worden weergegeven. Wijzigingen in de interface van de Reader-werkbalk (zoals in besturingselementen voor positie, ankerpunten en zichtbaarheid) blijven van toepassing als u PDF-bestanden vervolgens weergeeft in zowel AIR-toepassingen als de browser.

Belangrijk: *Om PDF-inhoud in AIR te renderen, moet Adobe Reader of Adobe® Acrobat® versie 8.1 of hoger op de computer van de gebruiker zijn geïnstalleerd.*

PDF-mogelijkheden detecteren

Adobe AIR 1.0 of hoger

Als Adobe Reader of Adobe Acrobat 8.1 of hoger niet op de computer van de gebruiker is geïnstalleerd, wordt PDF-inhoud niet weergegeven in een AIR-toepassing. Om na te gaan of een gebruiker PDF-inhoud kan renderen, moet u eerst de eigenschap `HTMLLoader.pdfCapability` controleren. Deze eigenschap is ingesteld op een van de volgende constanten uit de klasse HTMLPDFCapability.

Constante	Beschrijving
HTMLPDFCapability.STATUS_OK	Een geschikte versie (8.1 of hoger) van Adobe Reader is gedetecteerd en PDF-inhoud kan in een HTMLLoader-object worden geladen.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	Geen versie van Adobe Reader gedetecteerd. HTMLLoader-objecten kunnen geen PDF-inhoud weergeven.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Adobe Reader is gedetecteerd maar de versie is te oud. HTMLLoader-objecten kunnen geen PDF-inhoud weergeven.
HTMLPDFCapability.ERROR_PREFERRED_READER_TOO_OLD	Een geschikte versie (8.1 of hoger) van Adobe Reader is gedetecteerd maar de versie van Adobe Reader die is ingesteld om PDF-inhoud af te handelen, is ouder dan Reader 8.1. HTMLLoader-objecten kunnen geen PDF-inhoud weergeven.

Als de gebruiker een Windows-computer heeft waarop Adobe Acrobat of Adobe Reader versie 7.x of hoger wordt uitgevoerd, wordt die versie gebruikt, zelfs als een recentere versie is geïnstalleerd die het laden van PDF-inhoud ondersteunt. Als in dit geval de waarde van de eigenschap `pdfCapability` is ingesteld op `HTMLPDFCapability.STATUS_OK` terwijl een AIR-toepassing PDF-inhoud probeert te laden, geeft de oudere versie van Acrobat of Reader een waarschuwing weer (er wordt geen uitzondering gegenereerd in de AIR-toepassing). Als deze situatie zich bij uw eindgebruikers kan voordoen, kunt u instructies opgeven om Acrobat te sluiten terwijl uw toepassing wordt uitgevoerd. U wordt aangeraden deze instructies weer te geven als de PDF-inhoud niet binnen een aanvaardbare tijd wordt geladen.

Bij Linux zoekt AIR naar Adobe Reader op het pad (PATH) dat door de gebruiker is geëxporteerd (als het de opdracht acroread bevat) en in de map /opt/Adobe/Reader.

Aan de hand van de volgende code kan worden vastgesteld of een gebruiker PDF-inhoud in een AIR-toepassing kan weergeven. Als de gebruiker PDF-inhoud niet kan weergeven, wordt er door de code een foutcode getraceerd die overeenkomt met het HTMLPDFCapability-foutobject:

```
if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

PDF-inhoud laden

Adobe AIR 1.0 of hoger

U kunt een PDF toevoegen aan een AIR-toepassing door een HTMLLoader-instantie te maken, de afmetingen ervan in te stellen en het pad van een PDF te laden.

In het volgende voorbeeld ziet u hoe u een PDF vanaf een externe site laadt. Vervang de URLRequest door het pad naar een beschikbare externe PDF.

```
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);
```

U kunt ook inhoud laden vanaf bestands-URL's en AIR-specifieke URL-schema's, zoals app en app-storage. Voorbeeld: de volgende code laadt het bestand test.pdf naar de PDF-submap van de toepassingsmap:

app:/js_api_reference.pdf

Zie “URI-schema's” op pagina 860 voor meer informatie over URL-schema's van AIR.

Scripting van PDF-inhoud

Adobe AIR 1.0 of hoger

U kunt JavaScript gebruiken om PDF-inhoud te besturen, net zoals in een webpagina in de browser.

JavaScript-extensies voor Acrobat bieden onder andere de volgende mogelijkheden:

- Paginanavigatie en vergroting regelen
- Formulieren in het document verwerken
- Multimediagebeurtenissen besturen

Meer informatie over JavaScript-extensies voor Adobe Acrobat vindt u in Adobe Acrobat Developer Connection op http://www.adobe.com/devnet/acrobat/javascript.html.

Basisinformatie over PDF-communicatie in HTML

Adobe AIR 1.0 of hoger

JavaScript in een HTML-pagina kan een bericht naar JavaScript in PDF-inhoud versturen door de methode `postMessage()` op te roepen van het DOM-object dat de PDF-inhoud voorstelt. U hebt bijvoorbeeld de volgende ingesloten PDF-inhoud:

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

De volgende JavaScript-code in de bevattende HTML-inhoud verstuurt een bericht naar de JavaScript-code in het PDF-bestand:

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

Het PDF-bestand kan JavaScript bevatten om dit bericht te ontvangen. U kunt in bepaalde contexten JavaScript-code aan PDF-bestanden toevoegen, zoals context op document-, map-, pagina-, veld- en batchniveau. Alleen context op documentniveau, die scripts definieert die worden geëvalueerd wanneer het PDF-document wordt geopend, wordt hier beschreven.

Een PDF-bestand kan de eigenschap `messageHandler` toevoegen aan het `hostContainer`-object. De eigenschap `messageHandler` is een object dat de afhandelingsfuncties definieert waarmee op berichten wordt gereageerd. Voorbeeld: de volgende code definieert de functie om berichten af te handelen die het PDF-bestand ontvangt van de hostcontainer (dit is de HTML-inhoud die is ingesloten in het PDF-bestand):

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

JavaScript-code in de HTML-pagina kan de methode `postMessage()` oproepen van het PDF-object dat zich in de pagina bevindt. Door het oproepen van deze methode wordt het bericht `"Hello from HTML"` naar de JavaScript-code op documentniveau in het PDF-bestand verzonden:

```
<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>
```

Zie Cross-scripting van PDF-inhoud in Adobe AIR voor een meer geavanceerd voorbeeld en voor informatie over het gebruik van Acrobat 8 om JavaScript aan een PDF-bestand toe te voegen.

Scripting van PDF-inhoud vanuit ActionScript

Adobe AIR 1.0 of hoger

ActionScript-code (in SWF-inhoud) kan niet rechtstreeks communiceren met JavaScript in PDF-inhoud. ActionScript kan echter wel communiceren met de JavaScript-code in de HTML-pagina die is geladen in het HTMLLoader-object waarmee de PDF-inhoud wordt geladen, en die JavaScript-code kan communiceren met de JavaScript-code in het geladen PDF-bestand. Zie “HTML en JavaScript programmeren in AIR” op pagina 1039 voor meer informatie.

Bekende beperkingen voor PDF-inhoud in AIR

Adobe AIR 1.0 of hoger

PDF-inhoud in Adobe AIR is onderworpen aan de volgende beperkingen:

- PDF-inhoud wordt niet weergegeven in een venster (een NativeWindow-object) dat transparant is (waarbij de eigenschap `transparent` is ingesteld op `true`).
- De weergavevolgorde van een PDF-bestand is anders dan bij andere weegaveobjecten in een AIR-toepassing. Hoewel PDF-inhoud correct wordt weergegeven volgens de HTML-weergavevolgorde, bevindt deze zich altijd boven inhoud in de weergavevolgorde van de AIR-toepassing.

- Als bepaalde visuele eigenschappen van een HTMLLoader-object dat een PDF-document bevat worden gewijzigd, wordt het PDF-document onzichtbaar. Het betreft hierbij onder andere de eigenschappen `filters`, `alpha`, `rotation` en `scaling`. Als deze eigenschappen worden gewijzigd, wordt de PDF-inhoud onzichtbaar totdat de eigenschappen opnieuw worden ingesteld. De PDF-inhoud is eveneens onzichtbaar als u deze eigenschappen wijzigt bij weergaveobjectcontainers die het HTMLLoader-object bevatten.
- PDF-inhoud is alleen zichtbaar wanneer de eigenschap `scaleMode` van het Stage-object van het NativeWindow-object dat de PDF-inhoud bevat, is ingesteld op `StageScaleMode.NO_SCALE`. Wanneer deze eigenschap is ingesteld op een andere waarde, is de PDF-inhoud niet zichtbaar.
- Wanneer de gebruiker op koppelingen naar inhoud in het PDF-bestand klikt, wordt de schuifpositie van de PDF-inhoud bijgewerkt. Wanneer de gebruiker op koppelingen naar inhoud buiten het PDF-bestand klikt, wordt het HTMLLoader-object omgeleid waarin de PDF-inhoud zich bevindt (zelfs als het doel van een koppeling een nieuw venster is).
- Workflows voor PDF-commentaar werken niet in AIR.

Hoofdstuk 29: Basis van gebruikersinteractie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassing kan interactiviteit creëren door met ActionScript 3.0 op gebruikersactiviteit te reageren. Let op: deze sectie veronderstelt dat u het gebeurtenismodel van ActionScript 3.0 al kent. Zie "Gebeurtenissen afhandelen" op pagina 133 voor meer informatie.

Gebruikersinvoer vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruikersinteractie via toetsenbord, muis, camera of een combinatie van deze apparaten, ligt ten grondslag aan interactiviteit. Voor het identificeren van en het reageren op gebruikersinteractie in ActionScript 3.0 staat het luisteren naar gebeurtenissen voorop.

De klasse InteractiveObject, een subklasse van de klasse DisplayObject, biedt de algemene structuur van benodigde gebeurtenissen en functionaliteit voor verwerking van gebruikersinteractie. U kunt niet rechtstreeks een instantie van de klasse InteractiveObject maken. Weergaveobjecten zoals SimpleButton, Sprite, TextField en diverse componenten van het Flash-programma voor het schrijven van programmacode en van Flex nemen daarentegen het gebruikersinteractiemodel van deze klasse over en hebben dus dezelfde structuur. Dit betekent dat de technieken die u leert en de code die u schrijft voor verwerking van gebruikersinteractie in een object dat is afgeleid van InteractiveObject, ook op alle andere objecten van toepassing zijn.

Belangrijke concepten en termen

Het is van belang dat u bekend bent met de volgende belangrijke termen op het gebied van gebruikersinteractie voordat u verdergaat:

Tekencode Een numerieke code die een teken uit de huidige tekenset vertegenwoordigt (gekoppeld aan het drukken op een toets van het toetsenbord). 'D' en 'd' hebben bijvoorbeeld verschillende tekencodes, ook al worden deze op een Nederlands toetsenbord met behulp van dezelfde toets gemaakt.

Contextmenu Het menu dat wordt weergegeven wanneer de gebruiker met de rechtermuisknop klikt of een bepaalde combinatie maakt met toetsenbord en muis. Opdrachten uit een contextmenu worden doorgaans rechtstreeks toegepast op het item waarop is geklikt. Een contextmenu voor een afbeelding kan bijvoorbeeld een opdracht bevatten waarmee de afbeelding in een apart venster wordt weergegeven en een opdracht waarmee de afbeelding wordt gedownload.

Focus De aanduiding dat een geselecteerd element actief is en het doel is van interactie via toetsenbord of muis.

Toetscode Een numerieke code die overeenkomt met een fysieke toets op het toetsenbord.

Meer Help-onderwerpen

InteractiveObject

Keyboard

Mouse

ContextMenu

Focus beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een interactief object kan via programmacode of door een gebruikershandeling focus krijgen. Als de eigenschap `tabEnabled` op `true` wordt ingesteld, kan de gebruiker focus bovendien van een object naar het volgende object verplaatsen door op de Tab-toets te drukken. De waarde van `tabEnabled` is standaard `false`, behalve in de volgende gevallen:

- De waarde van een object SimpleButton is `true`.
- Voor een invoertekstveld is de waarde `true`.
- Voor een object Sprite of MovieClip waarvoor `buttonMode` is ingesteld op `true`, is de waarde `true`.

In beide situaties kunt u een listener voor `FocusEvent.FOCUS_IN` of `FocusEvent.FOCUS_OUT` toevoegen om additionele gedragingen mogelijk te maken wanneer de focus wijzigt. Dit is met name nuttig voor tekstvelden en formulieren, maar kan ook worden gebruikt voor Sprites, filmclips of andere objecten die van de klasse InteractiveObject overerven. In het volgende voorbeeld wordt weergegeven hoe u doorlopende focus met de Tab-toets mogelijk maakt en hoe u op de hieruit resulterende focusgebeurtenis kunt reageren. In dit geval verandert de kleur van elk vierkant zodra het vierkant focus krijgt.

Opmerking: *Flash Professional gebruikt sneltoetsen om focus te beheren; om focusbeheer correct te simuleren, moeten SWF-bestanden worden getest in een browser of AIR in plaats van in 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);
}
```

Invoertypen vaststellen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Vanaf Flash Player 10.1 en Adobe AIR 2 kunt u de runtimeomgeving testen op ondersteuning voor specifieke invoertypen. Met ActionScript kunt u testen of het apparaat waarop het runtimeprogramma op dat moment wordt uitgevoerd:

- Ondersteuning biedt voor invoer met de pen of vinger (of geen ondersteuning biedt voor aanraakinvoer)
- Beschikt over een virtueel of fysiek toetsenbord voor de gebruiker (of juist geen enkel toetsenbord heeft).
- Een cursor weergeeft (als dit niet het geval is, werken functies niet die afhankelijk zijn van een cursor die boven een object wordt geplaatst).

De ActionScript-API's waarmee het type invoer wordt vastgesteld zijn onder andere:

- flash.system.Capabilities.touchscreenType-eigenschap: deze runtimewaarde geeft aan welk invoertype in de huidige omgeving wordt ondersteund.
- flash.system.TouchscreenType-klasse: een klasse met opsommingswaardeconstanten voor de eigenschap Capabilities.touchScreenType.
- flash.ui.Mouse.supportsCursor-eigenschap: deze runtimewaarde geeft aan of een permanente cursor al dan niet beschikbaar is.
- flash.ui.Keyboard.physicalKeyboardType-eigenschap: deze runtimewaarde geeft aan of een volledig fysiek toetsenbord beschikbaar is, of alleen een numeriek toetsenbord, of helemaal geen toetsenbord.
- flash.ui.KeyboardType-klasse: een klasse met nummerwaardeconstanten voor de eigenschap flash.ui.Keyboard.physicalKeyboardType.
- flash.ui.Keyboard.hasVirtualKeyboard-eigenschap: een runtimewaarde die aangeeft of een virtueel toetsenbord wordt geboden aan de gebruiker (ofwel in plaats van een fysiek toetsenbord of als extra toetsenbord, naast het fysieke toetsenbord).

Met de API's voor het vaststellen van het invoertype kunt u de functies van een apparaat volledig benutten of alternatieven bieden als bepaalde functies ontbreken. Deze API's zijn vooral handig voor het ontwikkelen van mobiele toepassingen en toepassingen die werken met aanraaktechnologieën. Als u bijvoorbeeld een interface hebt voor een mobiel apparaat met kleine knoppen voor een pen, kunt u een alternatief interface met grotere knoppen aanbieden voor gebruikers die met hun vingers willen invoeren. De volgende code is voor een toepassing met de functie createStylusUI() die één bepaalde set met gebruikersinterface-elementen toewijst die geschikt zijn voor peninvoer. Een andere functie, createTouchUI() genaamd, wijst een andere set met gebruikersinterface-elementen toe die geschikt is voor interactie via de vingers:

```
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();
}
```

Bij het ontwikkelen van toepassingen voor verschillende invoeromgevingen moet u rekening houden met de volgende compatibiliteitsgrafiek:

Omgeving	**supportsCursor**	**touchscreenType == FINGER**	**touchscreenType == STYLUS**	**touchscreenType == NONE**
Traditionele bureaubladcomputer	true	false	false	true
Apparaten met capacitieve aanraakschermen (tablets, PDA's en telefoons die subtiele aanrakingen kunnen detecteren, zoals de Apple iPhone of Palm Pre)	false	true	false	false
Apparaten met resistieve aanraakschermen (tablets, PDA's en telefoons die precieze, harde aanrakingen detecteren, zoals de HTC Fuze)	false	false	true	false
Apparaten met schermen die niet op aanrakingen reageren (telefoons en apparaten met geavanceerde functies waarop toepassingen worden uitgevoerd, maar die niet werken met aanraakschermen)	false	false	false	true

Opmerking: *Verschillende apparaatplatforms bieden ondersteuning voor allerlei combinaties van invoertypen. Gebruik deze grafiek als een algemene richtlijn.*

Hoofdstuk 30: Toetsenbordinvoer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassing kan toetsenbordinvoer vastleggen en erop reageren en kan een IME manipuleren, zodat gebruikers niet-ASCII-teksttekens in multibyte-talen kunnen invoeren. Let op: deze sectie veronderstelt dat u het gebeurtenismodel van ActionScript 3.0 al kent. Zie "Gebeurtenissen afhandelen" op pagina 133 voor meer informatie.

Zie "Invoertypen vaststellen" op pagina 588 voor informatie over het type toetsenbordondersteuning tijdens runtime (fysiek, virtueel, alfanumeriek of numeriek met 12-toetsen).

Met IME (Input Method Editor) kunnen gebruikers complexe tekens en symbolen intypen op een standaardtoetsenbord. Met de IME-klassen kunt u gebruikers toestaan om de functies van hun systeem-IME toe te passen in uw toepassingen.

Meer Help-onderwerpen

flash.events.KeyboardEvent

flash.system.IME

Toetsenbordinvoer vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt ervoor zorgen dat weergaveobjecten die het interactiemodel overerven van de klasse InteractiveObject, op toetsenbordgebeurtenissen reageren door gebeurtenislisteners te gebruiken. U kunt bijvoorbeeld een gebeurtenislistener in het werkgebied plaatsen om te luisteren naar en te reageren op toetsenbordinvoer. In de volgende code legt een gebeurtenislistener een toetsdruk vast, waarna de eigenschappen van de toetsnaam en de toetscode worden weergegeven:

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " +
event.charCode + ")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Sommige toetsen, zoals Ctrl, genereren wel gebeurtenissen, maar hebben geen glyph-aanduiding.

In het vorige codevoorbeeld wordt toetsenbordinvoer voor het hele werkgebied door de toetsenbordgebeurtenislistener vastgelegd. U kunt ook een gebeurtenislistener voor een specifiek weergaveobject in het werkgebied schrijven; deze gebeurtenislistener wordt geactiveerd wanneer het object focus heeft.

In het volgende voorbeeld worden toetsaanslagen alleen in het deelvenster Uitvoer weergegeven wanneer de gebruiker iets in de instantie TextField invoert. Als u Shift ingedrukt houdt, wordt de rand van TextField tijdelijk rood.

In deze code wordt aangenomen dat er een instantie TextField met de naam `tf` in het werkgebied bestaat.

```
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;
    }
}
```

De klasse TextField maakt ook melding van een gebeurtenis `textInput` waar u naar kunt luisteren wanneer een gebruiker tekst invoert. Zie "Tekstinvoer vastleggen" op pagina 399 voor meer informatie.

Opmerking: *In de AIR-runtime kan een toetsenbordgebeurtenis worden geannuleerd. In de Flash Player-runtime kan een toetsenbordgebeurtenis niet worden geannuleerd.*

Toetscodes en tekencodes

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt toegang krijgen tot de eigenschappen `keyCode` en `charCode` van een toetsenbordgebeurtenis om te bepalen welke toets is ingedrukt en vervolgens andere handelingen te activeren. De eigenschap `keyCode` is een numerieke waarde die overeenkomt met de waarde van een toets op het toetsenbord. De eigenschap `charCode` is de numerieke waarde van die toets in de huidige tekenset. (De standaardtekenset is UTF-8, die ASCII ondersteunt.)

Het voornaamste verschil tussen de toetscode en de tekenwaarden is het feit dat een toetscodewaarde voor een bepaalde toets op het toetsenbord staat (de 1 op het numerieke toetsenblok verschilt van de 1 op de bovenste rij, maar de toets waarmee '1' wordt gegenereerd is gelijk aan de toets waarmee '!' wordt gegenereerd), terwijl de tekenwaarde voor een bepaald teken staat (de tekens R en r zijn verschillend).

Opmerking: *Zie de klasse flash.ui.Keyboard in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de toewijzingen van toetsen aan de bijbehorende tekencodewaarden in ASCII.*

De toewijzingen tussen toetsen en de bijbehorende toetscodes zijn afhankelijk van het apparaat en het besturingssysteem. Gebruik daarom geen toetstoewijzingen om handelingen te activeren. Gebruik in plaats daarvan de vooraf gedefinieerde constante waarden die worden aangeboden via de klasse Keyboard ter verwijzing naar de gewenste eigenschappen `keyCode`. Gebruik in plaats van de toetstoewijzing voor Shift bijvoorbeeld de constante `Keyboard.SHIFT` (zoals getoond in het vorige codevoorbeeld).

Hogere prioriteit KeyboardEvent

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals ook bij andere gebeurtenissen het geval is, wordt de volgorde van toetsenbordgebeurtenissen bepaald door de hiërarchie van weergaveobjecten en niet door de volgorde waarin methoden `addEventListener()` in code worden toegewezen.

Stel bijvoorbeeld dat u een tekstveld met de naam `tf` in een filmclip met de naam `container` plaatst en een gebeurtenislistener voor een toetsenbordgebeurtenis aan beide instanties toevoegt, zoals getoond in het volgende voorbeeld:

```
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 + ")");
}
```

Aangezien zowel het tekstveld als de bovenliggende container een listener bevat, wordt de functie `reportKeyDown()` voor elke toetsaanslag in TextField tweemaal aangeroepen. Voor elke ingedrukte toets verzendt het tekstveld een gebeurtenis voordat de filmclip `container` een gebeurtenis verzendt.

Het besturingssysteem en de webbrowser verwerken toetsenbordgebeurtenissen vóór Adobe Flash Player of AIR. Als er bijvoorbeeld in Microsoft Internet Explorer op Ctrl+W wordt gedrukt, wordt het browservenster gesloten voordat een eventueel daarin opgenomen SWF-bestand een toetsenbordgebeurtenis verzendt.

De klasse IME gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse IME kunt u de invoermethode-editor (IME) van het besturingssysteem manipuleren binnen Flash Player of Adobe AIR.

Met ActionScript kunt u het volgende bepalen:

- Of een IME is geïnstalleerd op de computer van de gebruiker (`Capabilities.hasIME`)
- Of de IME is ingeschakeld of uitgeschakeld op de computer van de gebruiker (`IME.enabled`)
- De omzettingsmodus die door de huidige IME wordt gebruikt (`IME.conversionMode`)

U kunt een invoertekstveld koppelen aan een bepaalde IME-context. Door van het ene naar het andere invoerveld te gaan kunt u van IME wisselen, bijvoorbeeld tussen Hiragana (Japans), getallen op volledige breedte, getallen op halve breedte, directe invoer, enzovoort.

Met een IME kunnen gebruikers niet-ASCII-teksttekens typen in multibyte-talen zoals Chinees, Japans en Koreaans.

Raadpleeg de documentatie van het besturingssysteem waarvoor u de toepassing ontwikkelt voor meer informatie over werken met IME's. Raadpleeg de volgende websites voor aanvullende bronnen:

- http://www.msdn.microsoft.com/goglobal/

- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

Opmerking: *Als de IME niet actief is op de computer van de gebruiker, mislukken aanroepen naar andere IME-methoden of -eigenschappen dan* `Capabilities.hasIME`*. Wanneer u de IME handmatig hebt geactiveerd, werken volgende ActionScript-aanroepen naar IME-methoden en -eigenschappen naar behoren. Wanneer u bijvoorbeeld een Japanse IME gebruikt, moet u deze activeren voordat u een IME-methode of -eigenschap kunt aanroepen.*

Bepalen of een IME is geïnstalleerd en ingeschakeld

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voordat u één van de IME-methoden of -eigenschappen aanroept, moet u altijd controleren of er op de computer van de gebruiker op dat moment een IME is geïnstalleerd en ingeschakeld. De volgende code toont hoe u kunt controleren of er op de computer van de gebruiker een IME is geïnstalleerd en ingeschakeld voordat u methoden aanroept:

```
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.");
}
```

De bovenstaande code controleert eerst met de eigenschap `Capabilities.hasIME` of de gebruiker een IME heeft geïnstalleerd. Als deze eigenschap is ingesteld op `true`, controleert de code vervolgens met de eigenschap `IME.enabled` of de IME van de gebruiker op dit moment is ingeschakeld.

Bepalen welke IME-omzettingsmodus op dit moment is ingeschakeld

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u meertalige toepassingen bouwt, moet u soms kunnen bepalen welke omzettingsmodus actief is op het systeem van de gebruiker. De volgende code laat zien hoe u kunt controleren of een IME is geïnstalleerd op de computer van de gebruiker, en zo ja, welke IME-omzettingsmodus op dat moment actief is:

```
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.";
}
```

De bovenstaande code controleert eerst of op het systeem een IME is geïnstalleerd. Vervolgens wordt gecontroleerd welke omzettingsmodus de huidige IME gebruikt door de eigenschap `IME.conversionMode` te vergelijken met elke constante in de klasse IMEConversionMode.

De IME-omzettingsmodus instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u de omzettingsmodus van de IME van de gebruiker wijzigt, moet u ervoor zorgen dat de code is opgenomen in een blok `try..catch`. Hiermee voorkomt u dat er bij het instellen van de omzettingsmodus met de eigenschap `conversionMode` een fout optreedt als de IME de omzettingsmodus niet kan instellen. De volgende code laat zien hoe u een blok `try..catch` gebruikt wanneer u de eigenschap `IME.conversionMode` instelt:

```
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;
    }
}
```

De bovenstaande code maakt eerst een tekstveld, dat wordt gebruikt om een statusbericht voor de gebruiker weer te geven. Vervolgens activeert de code de IME, indien deze is geïnstalleerd, en stelt de omzettingsmodus in op Koreaans. Als er op de computer van de gebruiker geen Koreaanse IME is geïnstalleerd, genereert Flash Player of AIR een fout die wordt afgevangen door het blok `try..catch`. Het blok `try..catch` geeft de foutmelding weer in het eerder gemaakte tekstveld.

De IME uitschakelen voor bepaalde tekstvelden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het kan voorkomen dat u de IME van de gebruiker wilt uitschakelen op het moment dat de gebruiker tekens invoert. Zo wilt u bij een tekstveld dat alleen numerieke invoer accepteert, wellicht niet dat de IME de gegevensinvoer vertraagt.

Het volgende voorbeeld laat zien hoe u kunt luisteren naar de gebeurtenissen `FocusEvent.FOCUS_IN` en `FocusEvent.FOCUS_OUT` en afhankelijk van het resultaat de IME van de gebruiker kunt uitschakelen:

```
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;
    }
}
```

In dit voorbeeld worden twee invoertekstvelden gemaakt, `phoneTxt` en `nameTxt`, waarna twee gebeurtenislisteners worden toegevoegd aan het tekstveld `phoneTxt`. Wanneer de gebruiker de focus instelt op het tekstveld `phoneTxt`, wordt een gebeurtenis `FocusEvent.FOCUS_IN` verzonden en wordt de IME uitgeschakeld. Wanneer het tekstveld `phoneTxt` focus verliest, wordt de gebeurtenis `FocusEvent.FOCUS_OUT` verzonden en wordt de IME weer ingeschakeld.

Luisteren naar IME-compositiegebeurtenissen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

IME-compositiegebeurtenissen worden verzonden wanneer een compositietekenreeks wordt ingesteld. Als de IME van de gebruiker bijvoorbeeld ingeschakeld en actief is en de gebruiker een tekenreeks invoert in het Japans, wordt de gebeurtenis `IMEEvent.IME_COMPOSITION` verzonden zodra de gebruiker de compositietekenreeks selecteert. Als u wilt luisteren naar de gebeurtenis `IMEEvent.IME_COMPOSITION`, moet u een gebeurtenislistener toevoegen aan de statische eigenschap `ime` in de klasse System (`flash.system.System.ime.addEventListener(...)`), zoals getoond in het volgende voorbeeld:

```
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;
}
```

De bovenstaande code maakt twee tekstvelden en voegt deze toe aan het weergaveoverzicht. Het eerste tekstveld, `inputTxt`, is een invoertekstveld waarin de gebruiker Japanse tekst kan invoeren. Het tweede tekstveld, `outputTxt`, is een dynamisch tekstveld dat foutmeldingen weergeeft of een echo toont van de Japanse tekenreeks die de gebruiker invoert in het tekstveld `inputTxt`.

Virtuele toetsenborden

Flash Player 10.2 of hoger, AIR 2.6 of hoger

Mobiele apparaten, zoals telefoons en tablets, hebben vaak een virtueel, elektronisch toetsenbord in plaats van een fysiek toetsenbord. Met de klassen in de Flash API kunt u:

- Detecteren wanneer het virtuele toetsenbord wordt weergegeven en gesloten.
- Voorkomen dat het toetsenbord wordt weergegeven.

- Bepalen welk gedeelte van het werkgebied in beslag wordt genomen door het virtuele toetsenbord.
- Interactieve objecten maken die het toetsenbord weergeven als ze de focus krijgen. (Niet ondersteund door AIR-toepassingen op iOS.)
- (alleen AIR) Schakel automatisch pannen uit, zodat de toepassing zelf de weergave kan aanpassen om ruimte te maken voor het toetsenbord.

Het virtuele toetsenbord besturen

De runtime geeft het virtuele toetsenbord automatisch weer wanneer een gebruiker een tekstveld of een speciaal geconfigureerd interactief object aanraakt. Wanneer het toetsenbord verschijnt, volgt de runtime de conventies van het native platform voor het pannen en aanpassen van de grootte van de toepassingsinhoud, zodat de gebruiker de tekst die hij of zij typt, kan zien.

Wanneer het toetsenbord verschijnt, verzendt het object dat de focus heeft achtereenvolgens de volgende gebeurtenissen:

De gebeurtenis `softKeyboardActivating` — wordt verzonden vlak voordat het toetsenbord in beeld komt in het werkgebied. Als u de methode `preventDefault()` van het verzonden gebeurtenisobject aanroept, wordt het virtuele toetsenbord niet weergegeven.

De gebeurtenis `softKeyboardActivate` — wordt verzonden nadat de gebeurtenis `softKeyboardActivating` is afgehandeld. Wanneer het object met focus deze gebeurtenis verzendt, wordt de eigenschap `softKeyboardRect` van het object Stage bijgewerkt om het gedeelte van het werkgebied dat wordt verborgen door het virtuele toetsenbord aan te geven. Deze gebeurtenis kan niet worden geannuleerd.

Opmerking: *Wanneer het formaat van het toetsenbord verandert, bijvoorbeeld wanneer de gebruiker het type toetsenbord wijzigt, verzendt het object dat de focus heeft een tweede gebeurtenis softKeyboardActivate.*

De gebeurtenis `softKeyboardDeactivate` — wordt verzonden wanneer het virtuele toetsenbord om welke reden dan ook wordt gesloten. Deze gebeurtenis kan niet worden geannuleerd.

Met het volgende voorbeeld voegt u twee TextField-objecten toe aan het werkgebied. Met het bovenste TextField voorkomt u dat het toetsenbord wordt weergegeven wanneer u op het veld tikt en sluit u het toetsenbord als het al wordt weergegeven. Het laagste TextField illustreert het standaardgedrag. Het voorbeeld rapporteert de gebeurtenissen met betrekking tot het elektronische toetsenbord die door beide tekstvelden worden verzonden.

```
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 );
    }
}
}
```

Virtueel toetsenbord ondersteunen voor interactieve objecten

Flash Player 10.2 en hoger, AIR 2.6 en hoger (maar niet ondersteund op iOS)

Normaliter wordt het virtuele toetsenbord alleen weergegeven wanneer een TextField-object wordt aangeraakt. U kunt een instantie van de InteractiveObject-klasse configureren, zodat het virtuele toetsenbord wordt geopend als het object de focus krijgt.

Stel de eigenschap `needsSoftKeyboard` van een InteractiveObject-instantie in op `true` om de instantie zodanig te configureren dat het elektronische toetsenbord wordt geopend. Wanneer het object wordt toegewezen aan de focuseigenschap van het werkgebied, wordt het elektronische toetsenbord automatisch geopend. Bovendien kunt u het toetsenbord weergeven door de methode `requestSoftKeyboard()` van het InteractiveObject aan te roepen.

Het volgende voorbeeld toont hoe u een InteractiveObject zodanig kunt programmeren dat het als een veld voor tekstinvoer fungeert. De klasse TextInput uit het voorbeeld stelt de eigenschap `needsSoftKeyboard` zodanig in dat het toetsenbord wordt weergegeven wanneer dat nodig is. Het object luistert vervolgens naar `keyDown`-gebeurtenissen en voegt het getypte teken in in het veld.

Het voorbeeld maakt gebruik van de Flash-tekstengine om getypte tekst toe te voegen en weer te geven en verwerkt enkele belangrijke gebeurtenissen. Om het eenvoudig te houden, implementeert het voorbeeld geen tekstveld met alle mogelijke functies.

```
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();
        }
    }
}
```

De volgende hoofdtoepassingsklasse toont hoe u de klasse TextInput kunt gebruiken en de toepassingslay-out kunt beheren wanneer het toetsenbord wordt weergegeven of de oriëntatie van het apparaat wordt gewijzigd. De hoofdklasse creëert een TextInput-object en stelt de grenzen van het object in om het werkgebied te vullen. De klasse past de grootte van het TextInput-object aan wanneer het virtuele toetsenbord verschijnt of wanneer de grootte van het werkgebied verandert. De klasse luistert naar elektronische-toetsenbordgebeurtenissen van het TextInput-object en naar vergroot- of verkleingebeurtenissen van het werkgebied. Ongeacht de oorzaak van de gebeurtenis bepaalt de toepassing het zichtbare gedeelte van het werkgebied en past het de grootte van het tekstinvoergebied aan het werkgebied aan. In een echte toepassing hebt u natuurlijk een meer geavanceerd lay-outalgoritme nodig.

```
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;
        }
    }
}
```

***Opmerking:** Het werkgebied verzendt alleen vergroot- of verkleingebeurtenissen als reactie op een wijziging van de oriëntatie wanneer de eigenschap `scaleMode` is ingesteld op `noScale`. In andere modi blijven de afmetingen van het werkgebied ongewijzigd en wordt de inhoud ter compensatie geschaald.*

Weergavewijzigingen in de toepassing verwerken

AIR 2.6 en hoger

In AIR kunt u de standaardfunctionaliteit voor pannen en vergroten/verkleinen uitschakelen door een virtueel toetsenbord weer te geven. Dat doet u door het element`softKeyboardBehavior`in het toepassingsbeschrijvingsbestand in te stellen op `geen`:

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Wanneer u de automatische functionaliteit uitschakelt, is het de taak van uw toepassing om eventuele benodigde wijzigingen aan te brengen in de weergave van de toepassing. Er wordt een softKeyboardActivate-gebeurtenis verzonden wanneer het toetsenbord wordt geopend. Wanneer de gebeurtenis `softKeyboardActivate` wordt verzonden, bevat de eigenschap `softKeyboardRect` van het werkgebied de afmetingen van het gebied dat door het geopende toetsenbord wordt verborgen. Gebruik deze afmetingen om de inhoud te verplaatsen of te vergroten/verkleinen, zodat deze goed wordt weergegeven wanneer het toetsenbord wordt geopend en de gebruiker aan het typen is. (Wanneer het toetsenbord niet wordt weergegeven, zijn de afmetingen van het softKeyboardRect-kader nul.)

Wanneer het toetsenbord wordt gesloten, wordt een gebeurtenis `softKeyboardDeactivate` verzonden en kunt u de standaardweergave van de toepassing herstellen.

```
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;
        }
    }
}
```

Opmerking: *Op Android zijn er enkele situaties, zoals de modus Volledig scherm, waarin de precieze afmetingen van het toetsenbord niet beschikbaar zijn vanuit het besturingssysteem. In deze gevallen wordt de grootte geschat. Bovendien wordt in de oriëntatie Liggend het native IME-toetsenbord van volledige grootte gebruikt voor de invoer van alle tekst. Dit IME-toetsenbord heeft een geïntegreerd veld voor tekstinvoer en beslaat het volledige werkgebied. Het is niet mogelijk een toetsenbord in de stand Liggend weer te geven dat niet het volledige scherm beslaat.*

Hoofdstuk 31: Muisinvoer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uw toepassing kan interactiviteit maken door het vastleggen en reageren op muisinvoer. Let op: deze sectie veronderstelt dat u het gebeurtenismodel van ActionScript 3.0 al kent. Zie "Gebeurtenissen afhandelen" op pagina 133 voor meer informatie.

Zie "Invoertypen vaststellen" op pagina 588 voor informatie over het type muisondersteuning tijdens runtime (permanente cursor, pen- of aanraakinvoer).

Meer Help-onderwerpen

flash.ui.Mouse

flash.events.MouseEvent

"Invoer via aanraken, multitouch en bewegingen" op pagina 613

Muisinvoer vastleggen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Muisklikken maken muisgebeurtenissen die kunnen worden gebruikt om interactieve functionaliteit te activeren. U kunt een gebeurtenislistener aan het werkgebied toevoegen om te luisteren naar muisgebeurtenissen die ergens in het SWF-bestand optreden. U kunt ook gebeurtenislisteners aan objecten in het werkgebied toevoegen die het interactiemodel van InteractiveObject overerven (bijvoorbeeld Sprite of MovieClip); deze listeners worden geactiveerd wanneer op het object wordt geklikt.

Net als toetsenbordgebeurtenissen, worden ook muisgebeurtenissen teruggekoppeld. In het volgende voorbeeld wordt de gebeurtenis zowel vanuit de sprite `square` als vanuit het object Stage verzonden wanneer er op square wordt geklikt, aangezien `square` een onderliggend object van Stage is:

```
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 + "]");
}
```

In het vorige voorbeeld bevat de muisgebeurtenis positiegegevens over de klik. De eigenschappen `localX` en `localY` bevatten de locatie van de klik op het item op het laagste niveau in de weergaveketen. Wanneer er bijvoorbeeld in de linkerbovenhoek van `square` wordt geklikt, worden de lokale coördinaten [0,0] gemeld, aangezien dit het registratiepunt van `square` is. De eigenschappen `stageX` en `stageY` verwijzen daarentegen naar de algemene coördinaten van de klik in het werkgebied. Bij dezelfde klik wordt [50,50] voor deze coördinaten gemeld, aangezien `square` naar deze coördinaten is verplaatst. Beide coördinatenparen kunnen nuttig zijn, al naar gelang de wijze waarop u op gebruikersinteractie wilt reageren.

Opmerking: *In de modus Volledig scherm kunt u de toepassing zodanig configureren dat muisvergrendeling wordt gebruikt. Bij de muisvergrendeling wordt de cursor uitgeschakeld en kan de muis zonder begrenzingen worden verplaatst. Zie "Werken met de modus Volledig scherm" op pagina 177 voor meer informatie.*

Het object MouseEvent bevat ook de Booleaanse eigenschappen `altKey`, `ctrlKey` en `shiftKey`. U kunt deze eigenschappen gebruiken om te controleren of er tijdens de muisklik ook op Alt, Ctrl of Shift wordt gedrukt.

Sprites door het werkgebied slepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Aan de hand van de methode `startDrag()` van de klasse Sprite kunt u toestaan dat gebruikers een Sprite-object door het werkgebied slepen. In de volgende code wordt hiervan een voorbeeld gegeven:

```
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);
}
```

Zie in "De positie wijzigen" op pagina 184 de sectie over het maken van muisinteractie voor slepen-en-neerzetten voor meer informatie.

Slepen en neerzetten in AIR

In Adobe AIR kunt u ondersteuning voor slepen en neerzetten inschakelen zodat gebruikers gegevens uit en naar uw toepassing kunnen slepen. Zie "Slepen en neerzetten in AIR" op pagina 642 voor meer informatie.

De muiscursor aanpassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De muiscursor (muisaanwijzer) kan worden verborgen of vervangen door een weergaveobject in het werkgebied. Als u de muiscursor wilt verbergen, roept u de methode `Mouse.hide()` aan. U kunt de cursor aanpassen door `Mouse.hide()` aan te roepen, in het werkgebied naar de gebeurtenis `MouseEvent.MOUSE_MOVE` te luisteren en de coördinaten van een weergaveobject (de aangepaste cursor) in te stellen op de eigenschappen `stageX` en `stageY` van de gebeurtenis. In het volgende voorbeeld wordt een eenvoudige uitvoering van deze taak weergegeven:

```
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;
}
```

Voorbeeld van muisinvoer: WordSearch

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Dit voorbeeld demonstreert gebruikersinteractie door middel van verwerking van muisgebeurtenissen. Gebruikers bouwen zoveel mogelijk woorden op uit een willekeurig raster met letters, waarbij zij zowel in horizontale als verticale richting woorden in het raster kunnen plaatsen, maar elke letter maar één keer kunnen gebruiken. Dit voorbeeld demonstreert de volgende functies van ActionScript 3.0:

- Raster van componenten dynamisch opbouwen
- Reageren op muisgebeurtenissen
- Score bijhouden op basis van gebruikersinteractie

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van Wordsearch vindt u in de map Samples/Wordsearch. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
WordSearch.as	De klasse die de hoofdfunctionaliteit van de toepassing biedt.
WordSearch.fla of WordSearch.mxml	Het hoofdtoepassingsbestand voor Flex (MXML) of Flash (FLA).
dictionary.txt	Een bestand dat wordt gebruikt om te bepalen of geplaatste woorden punten opleveren en juist zijn gespeld.

Woordenboek laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een spel wilt maken waarbij er naar woorden moet worden gezocht, is er een woordenboek nodig. Het voorbeeld omvat een tekstbestand met de naam dictionary.txt met een lijst met woorden, die van elkaar zijn gescheiden door enter-tekens. Nadat er een array met de naam `words` is gemaakt, wordt dit bestand opgevraagd door de functie `loadDictionary()` en zodra dit is geladen, wordt het bestand een lange tekenreeks. U kunt deze tekenreeks parseren tot een array van woorden door gebruik te maken van de methode `split()`, met een onderbreking na elk Enter-teken (tekencode 10) of na elke nieuwe regel (tekencode 13). Het parseren vindt plaats in de functie `dictionaryLoaded()`:

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

De gebruikersinterface maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Nadat de woorden zijn opgeslagen, kunt u de gebruikersinterface instellen. Maak twee instanties Button: één voor het verzenden van een woord en één voor het wissen van een woord dat momenteel wordt gespeld. In beide gevallen moet u op de gebruikersinvoer reageren door te luisteren naar de gebeurtenis `MouseEvent.CLICK` die door de knop wordt uitgezonden en vervolgens een functie aan te roepen. In de functie `setupUI()` maakt de volgende code de listeners op de twee knoppen:

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

Een spelbord genereren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het spelbord is een raster van willekeurige letters. In de functie `generateBoard()` wordt een tweedimensionaal raster gemaakt door een lus in een andere lus te nesten. De eerste lus verhoogt de rijen en de tweede luis verhoogt het totale aantal kolommen per rij. Elk van de cellen die door deze rijen en kolommen worden gemaakt, bevat een knop die een letter op het bord vertegenwoordigt.

```
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);
        }
    }
}
```

Hoewel er slechts op één regel een listener voor een gebeurtenis `MouseEvent.CLICK` wordt toegevoegd, wordt de listener aan elke instantie Button toegewezen, aangezien deze zich in een lus `for` bevindt. Bovendien wordt er aan elke knop een naam toegewezen die van de rij- en kolompositie is afgeleid, zodat er verderop in de code eenvoudig naar de rij en kolom van elke knop kan worden verwezen.

Woorden opbouwen op basis van gebruikersinvoer

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Woorden kunnen worden geplaatst door letters te selecteren die in horizontale of verticale richting aan elkaar grenzen, maar elke letter kan slechts één keer worden gebruikt. Bij elke klik wordt er een muisgebeurtenis gegenereerd, waarna het door de gebruiker geplaatste woord moet worden gecontroleerd om na te gaan of dit doorloopt na letters waarop eerder is geklikt. Als dit niet het geval is, wordt het eerdere woord verwijderd en wordt een nieuw woord begonnen. Deze controle vindt plaats in de methode `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));
}
```

De methoden `charAt()` en `indexOf()` van de klasse String halen de juiste rijen en kolommen op van de knop waarop momenteel wordt geklikt en van de knop waarop daarvoor is geklikt. De methode `isLegalContinuation()` retourneert `true` als de rij of kolom niet is gewijzigd en als de rij of kolom die is gewijzigd slechts één stap verschilt van de vorige. Als u de regels van het spel wilt wijzigen en diagonale plaatsing wilt toestaan, kunt u de controle op een ongewijzigde rij of kolom verwijderen en ziet de laatste regel er als volgt uit:

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol -
currButtonCol) <= 1));
```

Geplaatste woorden controleren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Ter afronding van de code voor het spel zijn er mechanismen ter controle van geplaatste woorden en voor het bijhouden van de score nodig. De methode `searchForWord()` voorziet in beide:

```
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;
}
```

Deze functie doorloopt alle woorden in het woordenboek. Als het woord van de gebruiker overeenkomt met een woord in het woordenboek, wordt de corresponderende positie in het woordenboek geretourneerd. De methode `submitWord()` controleert vervolgens het antwoord en werkt de score bij als de positie geldig is.

Aanpassingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Aan het begin van de klasse staan diverse constanten. U kunt dit spel aanpassen door deze variabelen te wijzigen. U kunt bijvoorbeeld de beschikbare speltijd wijzigen door de variabele `TOTAL_TIME` te verhogen. Ook kunt u de variabele `PERCENT_VOWELS` ietwat verhogen door de kans te verhogen dat er bestaande woorden gevormd kunnen worden.

Hoofdstuk 32: Invoer via aanraken, multitouch en bewegingen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

De afhandelingsfuncties voor aanraakgebeurtenissen van het Flash Platform kunnen onder andere worden toegepast bij invoer van een enkel contactpunt of voor meerdere contactpunten bij apparaten die aanraakgebeurtenissen ondersteunen. De Flash-runtimes verwerken bovendien gebeurtenissen waarbij meerdere aanraakpunten worden gecombineerd met verplaatsing, zodat er sprake is van een beweging. Met andere woorden: Flash-runtimes kunnen twee typen invoer interpreteren:

Aanraken invoer via een enkel contactpunt, zoals bij een vinger, pen of ander hulpmiddel op een aanraakapparaat. Sommige apparaten bieden ondersteuning voor meerdere contactpunten tegelijkertijd (bijvoorbeeld via een vinger of pen).

Multitouch invoer via meerdere contactpunten tegelijkertijd

Bewegingen Invoer die door een apparaat of besturingssysteem wordt geïnterpreteerd als reactie op een of meer aanraakgebeurtenissen. Bijvoorbeeld wanneer een gebruiker twee vingers gelijktijdig draait en het apparaat of het besturingssysteem deze aanraakinvoer interpreteert als een draaibeweging. Sommige bewegingen worden met één vinger of aanraakpunt uitgevoerd, andere met meerdere aanraakpunten. Het apparaat of besturingssysteem bepaalt welk type beweging wordt toegewezen aan de invoer.

Afhankelijk van het gebruikersapparaat kan zowel aanraak- als bewegingsinvoer worden beschouwd als multitouch-invoer. ActionScript biedt een API voor het afhandelen van aanraak- en bewegingsgebeurtenissen en van individueel vastgelegde aanraakgebeurtenissen voor multitouch-invoer.

Opmerking: *Luisteren naar aanraak- en gebaargebeurtenissen kan een aanzienlijke belasting zijn voor de verwerkingsbronnen (gelijk aan het renderen van meerdere frames per seconde), afhankelijk van het apparaat en het besturingssysteem. Het is vaak beter muisgebeurtenissen te gebruiken als u de door gebaren of aanraken toegevoegde functionaliteit niet nodig hebt. Wanneer u aanraak- of gebaargebeurtenissen gebruikt, kunt u het aantal mogelijke grafische wijzigingen reduceren, vooral als dergelijke gebeurtenissen snel verzonden kunnen worden, zoals tijdens pan-, roteer- of zoombewerkingen. U kunt bijvoorbeeld de animatie in een component stoppen terwijl de gebruiker het formaat van de component wijzigt met een zoomgebaar.*

Meer Help-onderwerpen

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

Mike Jones: Virtuele gamecontrollers

Basisprincipes van aanraakinvoer

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Wanneer het Flash Platform wordt uitgevoerd in een omgeving die ondersteuning biedt voor aanraakinvoer, kunnen InteractiveObject-instanties luisteren naar aanraakgebeurtenissen en handlers aanroepen. Over het algemeen worden aanraak-, multitouch- en bewegingsgebeurtenissen op dezelfde manier afgehandeld als andere gebeurtenissen in ActionScript (zie "Gebeurtenissen afhandelen" op pagina 133 voor algemene informatie over het afhandelen van gebeurtenissen met ActionScript).

Voor een correcte interpretatie van een aanraking of beweging moet de Flash-runtime worden uitgevoerd in een hardware- en softwareomgeving die ondersteuning biedt voor aanraak- of multitouch-invoer. Zie "Invoertypen vaststellen" op pagina 588 voor een grafiek waarin verschillende typen aanraakschermen worden vergeleken. Als de runtime wordt uitgevoerd in een containertoepassing (zoals een browser), geldt bovendien dat die container de invoer doorgeeft aan de runtime. In sommige gevallen wordt multitouch weliswaar ondersteund door de huidige hardware en het besturingssysteem, maar wordt de invoer geïnterpreteerd door de browser waarin de Flash-runtime wordt uitgevoerd, en niet doorgegeven aan de Flash-runtime. Het komt ook voor dat de invoer wordt genegeerd.

Het volgende diagram laat zien hoe de invoer wordt doorgegeven van de gebruiker naar de runtime:

Invoerstroom van gebruiker naar de Flash Platform-runtime

Gelukkig beschikt de ActionScript-API voor het ontwikkelen van aanraaktoepassingen over klassen, methoden en eigenschappen waarmee de ondersteuning voor aanraak- en multitouch-invoer in runtimeomgevingen kan worden vastgesteld. De API waarmee u de ondersteuning voor aanraakinvoer kunt vaststellen is de API voor detectie voor aanraakgebeurtenisafhandeling.

Belangrijke concepten en termen

De volgende woordenlijst bevat belangrijke termen voor het schrijven van toepassingen voor het afhandelen van aanraakgebeurtenissen:

API voor detectie De methoden en eigenschappen waarmee de runtimeomgeving kan worden getest op ondersteuning voor aanraakgebeurtenissen en andere invoermodi.

Aanraakgebeurtenis Een handeling die wordt ingevoerd op een aanraakapparaat via een enkel contactpunt.

Aanraakpunt Het contactpunt voor een enkele aanraakgebeurtenis. Zelfs als een apparaat geen ondersteuning biedt voor bewegingsinvoer, kan het apparaat ondersteuning bieden voor meerdere gelijktijdige aanraakpunten.

Aanraakreeks De reeks gebeurtenissen die staat voor de levensduur van een enkele aanraking.. Deze gebeurtenissen omvatten één beginpunt, nul of meer verplaatsingen en één eindpunt.

Multitouch-gebeurtenis Een handeling die wordt ingevoerd op een aanraakapparaat met meerdere contactpunten (bijvoorbeeld met meer vingers tegelijkertijd).

Bewegingsgebeurtenis Een handeling die wordt ingevoerd op een aanraakapparaat waarbij een complexe beweging wordt getraceerd. Een voorbeeld van een beweging is wanneer u een scherm met twee vingers aanraakt, waarna u uw vingers gelijktijdig verplaatst in de vorm van een abstracte cirkel om een rotatie aan te geven.

Fasen Afzonderlijke tijdstippen in de gebeurtenisstroom (zoals het begin- en eindpunt)

Pen Een instrument voor interactie met een aanraakscherm. Een pen werkt preciezer dan een vinger. Sommige apparaten herkennen alleen invoer van een bepaald type pen. Apparaten die peninvoer herkennen hoeven niet per se invoer via meerdere contactpunten of met meerdere vingers tegelijkertijd te herkennen.

Druk- en tikbeweging Een specifiek type multitouch-invoerbeweging waarbij de gebruiker één vinger tegen het aanraakapparaat drukt en vervolgens met een andere vinger of aanwijsapparaat tikt. Deze beweging wordt vaak gebruikt om het rechtsklikken bij de muis te simuleren bij multitouch-toepassingen.

De API-structuur voor aanraakinvoer

De ActionScript-API voor aanraakinvoer is ontworpen met het oog op het feit dat het afhandelen van aanraakinvoer afhankelijk is van de hardware- en softwareomgeving van de Flash-runtime. De API biedt een primaire aanpak van drie behoeften bij de ontwikkeling van aanraaktoepassingen: detectie, gebeurtenissen en fasen. De coördinatie van deze API biedt een voorspelbare en gewenste ervaring voor de gebruiker, zelfs als u het doelapparaat niet kent tijdens de ontwikkeling van uw toepassing.

Detectie

De API voor detectie biedt de mogelijkheid om de hardware- en softwareomgeving tijdens runtime te testen. De waarden die door de runtime worden ingevuld bepalen welke aanraakinvoer beschikbaar is voor de Flash-runtime in de huidige context. Met de verzameling detectie-eigenschappen en -methoden kunt u de toepassing laten reageren op muisgebeurtenissen (in plaats van aanraakgebeurtenissen voor het geval dat sommige aanraakinvoer niet wordt ondersteund door de omgeving). Zie “Detectie voor aanraakondersteuning” op pagina 616 voor meer informatie.

Gebeurtenissen

ActionScript beheert aanraakgebeurtenissen met behulp van gebeurtenislisteners en -handlers, net als bij andere gebeurtenissen. Bij aanraakgebeurtenissen moet u echter ook rekening houden met het volgende:

- Een aanraking kan op meerdere manieren worden geïnterpreteerd door een apparaat of besturingssysteem, bijvoorbeeld als een reeks aanrakingen of, collectief, als een beweging.
- Bij een enkele aanraking op een aanraakapparaat (met een vinger, pen of aanwijsapparaat) wordt altijd ook een muisgebeurtenis verzonden. U kunt de muisgebeurtenis afhandelen met de gebeurtenistypen in de MouseEvent-klasse. Of u kunt uw toepassing zo ontwerpen dat deze alleen reageert op aanraakgebeurtenissen. U kunt de toepassing natuurlijk ook ontwerpen dat op beide gebeurtenistypen wordt gereageerd.
- Een toepassing kan reageren op meerdere gelijktijdige aanraakgebeurtenissen en elke gebeurtenis afzonderlijk afhandelen.

Gebruik de API voor detectie vooral om de gebeurtenissen die door uw toepassing worden afgehandeld, conditioneel af te handelen, en ook om te bepalen hoe de gebeurtenissen worden afgehandeld. Nadat de runtimeomgeving is verkend door de toepassing, wordt de juiste handler aangeroepen of wordt bepaald wat het juiste gebeurtenisobject is voor de gebruikersinteractie met de toepassing. Het komt ook voor dat de toepassing aangeeft dat specifieke invoer niet kan worden afgehandeld in de huidige omgeving en dat de gebruiker een alternatief krijgt aangeboden of hierover wordt geïnformeerd. Zie “Aanraakgebeurtenissen afhandelen” op pagina 617 en “Bewegingsgebeurtenissen afhandelen” op pagina 622 voor meer informatie.

Fasen

Bij aanraak- en multitouch-toepassingen bevatten aanraakgebeurtenisobjecten bepaalde eigenschappen waarmee de fasen van de gebruikersinteractie kunnen worden getraceerd. Schrijf ActionScript-code voor de afhandeling van fasen zoals de begin-, update- of eindfase van gebruikersinvoer. Zo kunt u de gebruikers voorzien van feedback. Laat het programma reageren op eindfasen, zodat visuele objecten veranderen als de gebruiker ze aanraakt en het aanraakpunt op het scherm verplaatst. U kunt de fasen ook gebruiken om bepaalde eigenschappen van een beweging bij te houden tijdens de uitvoering van de beweging.

Voor aanraakpuntgebeurtenissen kunt u bijhouden hoe lang de gebruiker 'stil blijft staan' bij een bepaald interactief object. Een toepassing kan alle fasen van een gebeurtenis met meerdere gelijktijdige aanraakpunten afzonderlijk bijhouden en overeenkomstig afhandelen.

Zorg ervoor dat u bij bewegingen ook de specifieke informatie over de transformatie van de beweging tijdens de uitvoering ervan kunt interpreteren. Houd de coördinaten van het contactpunt (of meerdere contactpunten) bij tijdens de verplaatsingen op het scherm.

Detectie voor aanraakondersteuning

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Met de eigenschappen van de Multitouch-klasse bepaalt u de mate waarin aanraakgebeurtenissen kunnen worden afgehandeld door uw toepassing. Vervolgens test u de omgeving om zeker te zijn van de benodigde ondersteuning voor de gebeurtenissen die worden afgehandeld door uw ActionScript-code. Eerst moet u bepalen welk type aanraakinvoer van belang is voor uw toepassing. De opties zijn: aanraakpunt, beweging of geen (in het laatste geval wordt alle aanraakinvoer alleen geïnterpreteerd als muisklikken en muisgebeurtenishandlers). Vervolgens gebruikt u de eigenschappen en methoden van de Multitouch-klasse om ervoor te zorgen dat de runtimeomgeving ondersteuning biedt voor het type aanraakinvoer dat wordt vereist door uw toepassing. Test de runtimeomgeving op ondersteuning van het type aanraakinvoer (worden bewegingen bijvoorbeeld geïnterpreteerd?) en handel op basis van deze situatie.

Opmerking: *De eigenschappen van de Multitouch-klasse zijn statisch en behoren niet tot instanties van een bepaalde klasse. Gebruik deze eigenschappen met de syntaxis Multitouch.property, zoals bijvoorbeeld:*

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

Het invoertype instellen

De Flash-runtime moet weten welk type aanraakinvoer moet worden geïnterpreteerd. Een aanraakgebeurtenis kan namelijk bestaan uit veel elementen of fasen. Als een vinger het aanraakscherm aanraakt, moet de runtime dan een aanraakgebeurtenis verzenden? Moet er misschien worden gewacht op een beweging? Of wordt de aanraking door de runtime bijgehouden als een muisklikgebeurtenis? Een toepassing die ondersteuning biedt voor aanraakinvoer moet kunnen bepalen welk type aanraakgebeurtenis moet worden afgehandeld voor de Flash-runtime. Met de eigenschap `Multitouch.inputMode` kunt u het type aanraakinvoer voor de runtime vaststellen. Er zijn drie invoermodi:

Geen Er wordt geen speciale afhandeling geboden voor aanraakgebeurtenissen. Stel het volgende in: `Multitouch.inputMode=MultitouchInputMode.NONE` en gebruik de MouseEvent-klasse om de invoer af te handelen.

Enkelvoudige aanraakpunten Alle aanraakinvoer wordt individueel geïnterpreteerd en alle aanraakpunten kunnen worden bijgehouden en afgehandeld. Stel het volgende in: `Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT` en gebruik de TouchEvent-klasse om de invoer af te handelen.

Bewegingsinvoer Het apparaat of besturingssysteem interpreteert de invoer als een complexe vorm van vingerbewegingen over het scherm. Het apparaat of besturingssysteem wijst de verplaatsing collectief toe aan een enkele bewegingsinvoergebeurtenis. Stel het volgende in:
`Multitouch.inputMode=MultitouchInputMode.GESTURE` en gebruik de klassen TransformGestureEvent, PressAndTapGestureEvent of GestureEvent om de invoer af te handelen.

Zie "Aanraakgebeurtenissen afhandelen" op pagina 617 voor een voorbeeld waarin het invoertype wordt ingesteld met de eigenschap `Multitouch.inputMode` voordat een aanraakgebeurtenis wordt afgehandeld.

Testen op ondersteuning voor aanraakinvoer

Andere eigenschappen van de Multitouch-klasse bieden waarden voor het afstemmen van uw toepassing op het type aanraakondersteuning in de huidige omgeving. De Flash-runtime vult waarden in voor het aantal gelijktijdige aanraakpunten dat is toegestaan of voor de beschikbare bewegingen. Als de runtime wordt uitgevoerd in een omgeving zonder ondersteuning voor de afhandeling van aanraakgebeurtenissen die door uw toepassing worden vereist, moet u de gebruiker een alternatief bieden. Zo kunt u bijvoorbeeld een afhandelingsfunctie bieden voor muisgebeurtenissen of de gebruiker informatie geven over welke functies wel en niet beschikbaar zijn in de huidige omgeving.

U kunt ook de API gebruiken voor ondersteuning van het toetsenbord, aanrakingen en de muis, zie "Invoertypen vaststellen" op pagina 588.

Zie "Problemen oplossen" op pagina 626 voor meer informatie over het testen op compatibiliteit.

Aanraakgebeurtenissen afhandelen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Basisaanraakgebeurtenissen worden in ActionScript op dezelfde manier afgehandeld als andere gebeurtenissen, zoals muisgebeurtenissen. U kunt luisteren naar een reeks aanraakgebeurtenissen die zijn gedefinieerd door de constanten voor gebeurtenistypen in de TouchEvent-klasse.

Opmerking: *Bij multitouch-puntinvoer (wanneer u bijvoorbeeld een aparaat met meer vingers aanraakt), worden bij de eerste aanraking een muisgebeurtenis en een aanraakgebeurtenis verzonden.*

Als u een basisaanraakgebeurtenis wilt afhandelen, gaat u als volgt te werk:

1. Configureer uw toepassing voor het afhandelen van aanraakgebeurtenissen door de eigenschap `flash.ui.Multitouch.inputMode` in te stellen op `MultitouchInputMode.TOUCH_POINT`.
2. Koppel een gebeurtenislistener aan een klasseninstantie die de eigenschappen van de InteractiveObject-klasse overneemt, zoals bijvoorbeeld Sprite of TextField.
3. Geef aan welk type aanraakgebeurtenis moet worden afgehandeld.
4. Roep een gebeurtenisafhandelingsfunctie aan om te reageren op de gebeurtenis.

Met de volgende code wordt een bericht weergegeven als het vierkant op mySprite wordt aangeraakt op een aanraakscherm.

```
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);
}
```

Eigenschappen van aanraakgebeurtenissen

Wanneer een gebeurtenis optreedt, wordt een gebeurtenisobject gemaakt. Het TouchEvent-object bevat informatie over de locatie en voorwaarden van de aanraakgebeurtenis. Met de eigenschappen van het gebeurtenisobject kunt u deze informatie ophalen.

Met de volgende code wordt het TouchEvent-object *evt* gemaakt, waarna de eigenschap `stageX` van het gebeurtenisobject in het tekstveld wordt weergegeven (de x-coördinaat van het punt in het werkgebied waar de aanraking plaatsvond):

```
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);
}
```

Zie de TouchEvent-klasse voor de eigenschappen die beschikbaar zijn via het gebeurtenisobject.

Opmerking: *Niet alle TouchEvent-eigenschappen worden in alle runtimeomgevingen ondersteund. Zo kan niet bij alle aanraakapparaten worden bepaald hoeveel druk door de gebruiker op het aanraakscherm wordt toegepast. Op deze apparaten wordt de eigenschap `TouchEvent.pressure` dan ook niet ondersteund. Test of specifieke eigenschappen worden ondersteund. Zo weet u zeker dat uw toepassing werkt. Ga naar "Problemen oplossen" op pagina 626 voor meer informatie.*

Fasen voor aanraakgebeurtenissen

Zorg ervoor dat aanraakgebeurtenissen in verschillende fasen worden bijgehouden, ook buiten een InteractiveObject, net als bij muisgebeurtenissen. Zorg er ook voor dat aanraakgebeurtenissen aan het begin, midden en einde van een aanraakinteractie wordt bijgehouden. De TouchEvent-klasse biedt waarden voor het afhandelen van de gebeurtenissen `touchBegin`, `touchMove` en `touchEnd`.

Zo kunt u met de gebeurtenissen `touchBegin`, `touchMove` en `touchEnd` visuele feedback geven aan de gebruiker wanneer hij een weergaveobject aanraakt en verplaatst:

```
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;
}
```

Opmerking: *De initiële aanraaklistener wordt gekoppeld aan mySprite, maar dit geldt niet voor de listeners voor het verplaatsen en beëindigen van de aanraakgebeurtenis. Als de vinger of het aanraakapparaat van de gebruiker zich vooruitbeweegt vóór het weergaveobject, blijft het werkgebied luisteren naar de aanraakgebeurtenis.*

Aanraakpunt-id

De eigenschap `TouchEvent.touchPointID` is een essentieel element bij het schrijven van toepassingen die reageren op aanraakinvoer. De Flash-runtime wijst een unieke `touchPointID`-waarde toe aan elk aanraakpunt. Wanneer een toepassing reageert op de fasen of verplaatsing van een aanraakinvoer, moet u de `touchPointID` controleren voordat de gebeurtenis wordt afgehandeld. De sleepmethoden voor aanraakinvoer van de Sprite-klasse gebruiken de eigenschap `touchPointID` als parameter, zodat de juiste invoerinstantie wordt afgehandeld. De eigenschap `touchPointID` zorgt ervoor dat een gebeurtenishandler reageert op het juiste aanraakpunt. Als dit niet het geval was, zou de gebeurtenishandler reageren op elke instantie van het aanraakgebeurtenistype op het apparaat (zoals alle `touchMove`-gebeurtenissen), wat zou leiden tot onvoorspelbaar gedrag. De eigenschap is vooral belangrijk wanneer de gebruiker objecten sleept.

Met de eigenschap `touchPointID` kunt u een gehele aanraakreeks beheren. Een aanraakreeks heeft één `touchBegin`-gebeurtenis, nul of meer `touchMove`-gebeurtenissen en één `touchEnd`-gebeurtenis, alle met dezelfde `touchPointID`-waarde.

In het volgende voorbeeld wordt een variabele *touchMoveID* vastgelegd om op de juiste `touchPointID`-waarde te testen voordat op een aanraakgebeurtenis wordt gereageerd. Anders wordt de gebeurtenishandler ook door andere aanraakinvoer geactiveerd. Zoals u ziet, bevinden de listeners voor de bewegings- en eindfase zich in het werkgebied, niet in het weergaveobject. Het werkgebied luistert naar de verplaatsings- of eindfase voor het geval dat de aanraking van de gebruiker verder gaat dan de grenzen van het weergaveobject.

```
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;
}
```

Aanraken en slepen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Er zijn twee methoden toegevoegd aan de Sprite-klasse om aanvullende ondersteuning te bieden voor toepassingen met aanraakinvoer: `Sprite.startTouchDrag()` en `Sprite.stopTouchDrag()`. Deze methoden gedragen zich hetzelfde als `Sprite.startDrag()` en `Sprite.stopDrag()` bij muisgebeurtenissen. Bij de twee methoden `Sprite.startTouchDrag()` en `Sprite.stopTouchDrag()` worden echter ook `touchPointID`-waarden geaccepteerd als parameter.

De runtime wijst de `touchPointID`-waarde toe aan het gebeurtenisobject voor een aanraakgebeurtenis. Met deze waarde kunt u reageren op een specifiek aanraakpunt wanneer de omgeving ondersteuning biedt voor meerdere gelijktijdige aanraakpunten (ook in het geval dat bewegingen niet worden afgehandeld). Voor meer informatie over de eigenschap `touchPointID` gaat u naar "Aanraakpunt-id" op pagina 619.

De volgende code bevat een eenvoudige start-sleep-gebeurtenishandler en een stop-sleepgebeurtenishandler voor een aanraakgebeurtenis. De variabele *bg* is een weergaveobject dat *mySprite* bevat:

```
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");
}
```

Hier vindt u een meer geavanceerd voorbeeld waarin een sleephandeling wordt gecombineerd met aanraakgebeurtenisfasen:

```
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;
}
```

Bewegingsgebeurtenissen afhandelen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Bewegingsgebeurtenissen worden op dezelfde manier afgehandeld als basisaanraakgebeurtenissen. U kunt luisteren naar een reeks bewegingsgebeurtenissen die worden gedefinieerd door de constanten voor het gebeurtenistype in de TransformGestureEvent-klasse, de GestureEvent -klasse en de PressAndTapGestureEvent-klasse.

Als u een bewegingsgebeurtenis wilt afhandelen, gaat u als volgt te werk:

1. Configureer uw toepassing voor het afhandelen van bewegingsinvoer door de eigenschap `flash.ui.Multitouch.inputMode` in te stellen op `MultitouchInputMode.GESTURE`.
2. Koppel een gebeurtenislistener aan een klasseninstantie die de eigenschappen van de InteractiveObject-klasse overneemt, zoals bijvoorbeeld Sprite of TextField.
3. Geef aan welk type bewegingsgebeurtenis moet worden afgehandeld.
4. Roep een gebeurtenisafhandelingsfunctie aan om te reageren op de gebeurtenis.

Met de volgende code wordt een bericht weergegeven als het vierkant op *mySprite* wordt weggezwiept op een aanraakscherm.

```
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);
}
```

Tikgebeurtenissen met twee vingers worden op dezelfde manier afgehandeld, maar maken wel gebruik van de GestureEvent-klasse:

```
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);
}
```

Druk- en tikgebeurtenissen worden ook op dezelfde manier afgehandeld, maar hierbij wordt gebruik gemaakt van de PressAndTapGestureEvent-klasse:

```
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);
}
```

Opmerking: *Niet alle GestureEvent-, TransformGestureEvent- en PressAndTapGestureEvent-gebeurtenistypen worden in alle runtimeomgevingen ondersteund. Een zwiepbeweging met meerdere vingers wordt bijvoorbeeld niet herkend door alle aanraakapparaten. Op deze apparaten worden InteractiveObject* `gestureSwipe`*-gebeurtenissen daarom ook niet ondersteund. Test of specifieke gebeurtenissen worden ondersteund. Zo weet u zeker dat uw toepassing werkt. Ga naar "Problemen oplossen" op pagina 626 voor meer informatie.*

Eigenschappen van bewegingsgebeurtenissen

Bewegingsgebeurtenissen hebben een kleiner bereik aan gebeurteniseigenschappen dan basisaanraakgebeurtenissen. Deze gebeurtenissen zijn op dezelfde manier toegankelijk, namelijk via het gebeurtenisobject in de gebeurtenisafhandelingsfunctie.

Met de volgende code wordt *mySprite* geroteerd wanneer de gebruiker er een draaibeweging op uitvoert. Het tekstveld laat zien hoeveel het object is gedraaid sinds de laatste beweging (wanneer u deze code test, kunt u het object enkele keren draaien om de kijken hoe de waarden veranderen):

```
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();
}
```

Opmerking: *Niet alle TransformGestureEvent -eigenschappen worden in alle runtimeomgevingen ondersteund. Een draaibeweging wordt bijvoorbeeld niet herkend door alle aanraakapparaten. Op deze apparaten wordt de eigenschap* `TransformGestureEvent.rotation` *dan ook niet ondersteund. Test of specifieke eigenschappen worden ondersteund. Zo weet u zeker dat uw toepassing werkt. Ga naar "Problemen oplossen" op pagina 626 voor meer informatie.*

Bewegingsfasen

Bovendien kunnen de bewegingsgebeurtenissen worden bijgehouden tijdens de verschillende fasen, zodat u de eigenschappen tijdens de beweging kunt bijhouden. U kunt bijvoorbeeld de x-coördinaten bijhouden terwijl een object wordt verplaatst met een zwiepbeweging. Met de bijgehouden waarden kunt u een lijn trekken door alle punten in het pad nadat de zwiepbeweging is voltooid. U kunt ook een object visueel wijzigen wanneer het object met een panbeweging over het scherm wordt gesleept. Wijzig het object nogmaals wanneer de panbeweging is voltooid.

```
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;
}
}
```

Opmerking: *De frequentie van de updatefase is afhankelijk van de runtimeomgeving. Bij sommige besturingssystemen en hardwarecombinaties worden updates helemaal niet verzonden.*

Bewegingsfase “all” bij eenvoudige bewegingsgebeurtenissen

Bij sommige bewegingsgebeurtenisobjecten worden de individuele fasen van de bewegingsgebeurtenis niet bijgehouden. In plaats hiervan krijgt de fase-eigenschap van het gebeurtenisobject de waarde 'all'. Bij de eenvoudige zwiepbeweging en het tikken met twee vingers worden de verschillende fasen van een gebeurtenis niet bijgehouden. De `phase`-eigenschap van het gebeurtenisobject van een InteractiveObject-luisterhandeling naar de gebeurtenissen `gestureSwipe` of `gestureTwoFingerTap` is altijd ingesteld op `all` wanneer de gebeurtenis wordt verzonden:

```
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"
}
```

Problemen oplossen

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

Technologieën voor hardware- en softwareondersteuning voor aanraakinvoer veranderen heel snel. Deze naslaggids bevat geen uitgebreide lijst met apparaten, besturingssystemen en softwarecombinaties waarbij multitouch-invoer wordt ondersteund. De gids geeft wel richtlijnen over het gebruik van de API voor detectie, zodat u kunt nagaan of uw toepassing wordt uitgevoerd op een apparaat dat ondersteuning biedt voor multitouch-invoer. Ook vindt u hier tips voor het oplossen van eventuele problemen met uw ActionScript-code.

Flash-runtimes reageren op aanraakgebeurtenissen op basis van de informatie die door het apparaat, het besturingssysteem of de containersoftware (zoals een browser) wordt doorgegeven aan de runtime. Door deze afhankelijkheid van de softwareomgeving wordt het lastig om precies de mate van multitouch-compatibiliteit vast te leggen. Sommige apparaten interpreteren een beweging of aanraakbeweging anders dan andere apparaten. Wordt rotatie gedefinieerd door twee vingers die tegelijkertijd draaien? Of wordt rotatie aangegeven door één vinger die een cirkel tekent? Afhankelijk van de hardware- en softwareomgeving kan de rotatiebeweging bestaan uit beide bewegingen, of uit iets heel anders. Het apparaat geeft de gebruikersinvoer door aan het besturingssysteem, dat deze informatie weer doorgeeft aan de runtime. Als de runtime is opgenomen in een browser, wordt de bewegings- of aanraakgebeurtenis soms geïnterpreteerd door de browsersoftware en niet doorgegeven aan de runtime. Dit gedrag is vergelijkbaar met het gedrag voor sneltoetsen: u probeert met een bepaalde toetsencombinatie een Flash Player-handeling binnen de browser uit te voeren, maar wat gebeurt is dat de browser een bepaald menu blijft openen.

Individuele API's en klassen geven aan als ze niet compatibel zijn met bepaalde besturingssystemen. Meer informatie over individuele API-invoer kunt u hier bekijken, te beginnen met de Multitouch-klasse: http://help.adobe.com/nl_NL/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html.

Hier volgen de beschrijvingen voor enkele algemene bewegingen en aanrakingen:

Pannen Een vinger verplaatsen van links naar rechts of omgekeerd. Bij sommige apparaten zijn twee vingers vereist.

Roteren Het scherm aanraken met twee vingers en deze vervolgens in een cirkel ronddraaien (alsof u met beide vingers een cirkel tekent op het scherm). Het draaipunt is het middelpunt tussen beide vingers.

Zwiepen Snel drie vingers verplaatsen van links naar rechts (of omgekeerd), of van boven naar beneden (of omgekeerd)

Zoomen Het scherm aanraken met twee vingers en deze vervolgens uit elkaar verplaatsen om in te zoomen, en naar elkaar toe om uit te zoomen.

Druk- en tikbeweging Met één vinger aanraken of verplaatsen en vervolgens tikken met een andere vinger.

Elk apparaat heeft eigen documentatie over de bewegingen die worden ondersteund en over hoe u elke beweging op het apparaat moet uitvoeren. Over het algemeen moet de gebruiker alle vingers van het apparaat afhalen tussen twee bewegingen. Dit is mede afhankelijk van het besturingssysteem.

Als uw toepassing niet reageert op aanraak- of bewegingsgebeurtenissen, moet u het volgende controleren:

1 Hebt u gebeurtenislisteners voor aanraak- of bewegingsgebeurtenissen gekoppeld aan een objectklasse die items overneemt van de InteractiveObject-klasse? Alleen met InteractiveObject-instanties kunt u luisteren naar aanraak- en bewegingsgebeurtenissen

2 Test u de toepassing binnen Flash Professional CS5? In dat geval moet u de toepassing publiceren en testen, aangezien Flash Professional de interactie kan onderscheppen.

3 Begin eenvoudig en kijk eerst naar wat wel functioneert (de volgende code komt uit de API-invoer voor `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);
}
```

Tik op de rechthoek. Als deze voorbeeldcode werkt, weet u zeker dat een enkele tikbeweging wordt ondersteund door de omgeving. Hierna kun u testen op het afhandelen van complexere bewegingen.

Het testen op bewegingsondersteuning is meer gecompliceerd. Individuele apparaten of besturingssystemen bieden ondersteuning voor geen enkele bewegingsinvoer, of voor willekeurige combinaties van bewegingsinvoer.

Hier is een eenvoudige test voor de in- en uitzoombeweging:

```
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";
}
```

Voer een zoombeweging uit op het apparaat en controleer of het bericht `Zoom is supported` (Zoomen wordt ondersteund) wordt weergegeven in het tekstveld. De gebeurtenislistener wordt toegevoegd aan het werkgebied, zodat u de beweging kunt uitvoeren in elk gedeelte van de testtoepassing.

Hier is een eenvoudige test voor de panbeweging:

```
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";
}
```

Voer een panbeweging uit op het apparaat en controleer of het bericht `Pan is supported` (Pannen wordt ondersteund) wordt weergegeven in het tekstveld. De gebeurtenislistener wordt toegevoegd aan het werkgebied, zodat u de beweging kunt uitvoeren in elk gedeelte van de testtoepassing.

Bij sommige combinaties van besturingssysteem en apparaat worden beide bewegingen ondersteund en bij andere combinaties geen van beide. Test de uitvoeringsomgeving van uw toepassing om zeker te zijn.

Bekende problemen

Hier volgt een lijst met bekende problemen gerelateerd aan aanraakinvoer:

1. Als u Mobile Internet Explorer uitvoert op het Windows Mobile-besturingssysteem wordt automatisch ingezoomd op de inhoud van een SWF-bestand:

   Dit zoomgedrag in Internet Explorer wordt overschreven als u de volgende code toevoegt aan de HTML-pagina die als host dient voor het SWF-bestand:

   ```
   <head>
   <meta name="viewport" content="width=device-width, height=device-height, initial-
   scale=1.0">
   </head>
   ```

2. Bij Windows 7 (en mogelijk ook bij andere besturingssystemen) moet de gebruiker het aanwijsapparaat (of de vingers) van het scherm optillen tussen twee afzonderlijke bewegingen. Als u een afbeelding wilt draaien en deze vervolgens wilt inzoomen, gaat u als volgt te werk:

   - Voer de draaibeweging uit.
   - Haal uw vingers van het scherm.
   - Plaats uw vingers weer op het scherm en voer de inzoombeweging uit.

3. Bij Windows 7 (en mogelijk ook bij andere besturingssystemen) worden de rotatie- en zoombewegingen niet altijd gevolgd door een updatefase als de gebruiker de beweging zeer snel uitvoert.

4. Multitouch wordt niet ondersteund op Windows 7 Starter Edition. Zie het AIR Labs Forum voor details: http://forums.adobe.com/thread/579180?tstart=0

5. Bij Mac OS 10.5.3 en hoger is de `Multitouch.supportsGestureEvents`-waarde altijd `true`, ook als de bewegingsgebeurtenissen niet worden ondersteund door de hardware.

Hoofdstuk 33: Kopiëren en plakken

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Gebruik de klassen in de klembord-API om informatie van en naar het systeemklembord te kopiëren. De gegevensindelingen die kunnen worden overgebracht van of naar een toepassing die wordt uitgevoerd in Adobe® Flash® Player of Adobe® AIR® zijn onder meer:

- Tekst
- Tekst met HTML-opmaak
- RTF-gegevens
- Geserialiseerde objecten
- Objectreferenties (alleen geldig in de oorspronkelijke toepassing
- Bitmaps (alleen AIR)
- Bestanden (alleen AIR)
- URL-tekenreeksen (alleen AIR)

Basisbeginselen van kopiëren en plakken

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

De kopiëren-en-plakken-API bevat de volgende klassen.

Pakket	Klassen
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

De statische eigenschap `Clipboard.generalClipboard` vertegenwoordigt het klembord van het besturingssysteem. De klasse Clipboard biedt methoden voor het lezen en schrijven van gegevens van en naar klembordobjecten.

De HTMLLoader-klasse (in AIR) en het standaardgedrag van de implementatie van de TextField-klasse voor de normale sneltoetsen voor kopiëren en plakken. Als u kopiëren en plakken in aangepaste componenten wilt gebruiken, kunt u rechtstreeks op deze toetsaanslagen wachten. U kunt ook native menuopdrachten met andere sneltoetsen creëren om onrechtstreeks op de toetsaanslagen te reageren.

U kunt in één klembordobject meerdere voorstellingen van dezelfde informatie beschikbaar maken om het andere toepassingen makkelijker te maken de gegevens te begrijpen en te gebruiken. Een afbeelding kan bijvoorbeeld als grafische gegevens, een geserialiseerd bitmapobject en een bestand worden opgenomen. Het omzetten van gegevens naar een indeling kan worden uitgesteld zodat de indeling pas wordt gecreëerd nadat de gegevens in de desbetreffende indeling zijn gelezen.

Lezen van en schrijven naar het systeemklembord

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Om het klembord van het besturingssysteem te lezen, roept u de `getData()`-methode aan van het `Clipboard.generalClipboard`-object en slaat u de naam van de indeling over om te lezen:

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

Opmerking: *Inhoud actief in Flash Player of in een niet-applicatie sandbox in AIR kan de `getData()`-methode alleen aanroepen in een gebeurtenishandler voor een `paste`-gebeurtenis. Met andere woorden kan alleen code die actief is in de AIR-toepassing de `getData()`-methode aanroepen buiten een `paste`-gebeurtenishandler.*

Als u naar het klembord wilt schrijven, voegt u de gegevens in een of meer indelingen toe aan het object `Clipboard.generalClipboard`. Bestaande gegevens in dezelfde indeling worden automatisch overschreven. het is echter een goede oefening om ook het systeemklembord leeg te maken, voordat u er nieuwe gegevens op schrijft, zodat ongerelateerde gegevens in andere indelingen ook worden verwijderd.

```
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);
```

Opmerking: *Actieve inhoud in Flash Player of in een niet-toepassingssandbox in AIR kan de `setData()`-methode alleen aanroepen in een gebeurtenishandler voor een gebruikersgebeurtenis, zoals een toetsenbord- of muisgebeurtenis of een `copy`- of `cut`-gebeurtenis. Met andere woorden, alleen actieve code in de AIR-toepassingssandbox kan de `setData()`-methode buiten een gebruikersgebeurtenishandler aanroepen.*

HTML kopiëren en plakken in AIR

Adobe AIR 1.0 of hoger

De HTML-omgeving in Adobe AIR biedt een eigen set gebeurtenissen en standaardgedrag voor kopiëren en plakken. Alleen code die in de toepassingssandbox wordt uitgevoerd, heeft rechtstreeks toegang tot het systeemklembord via het AIR-object `Clipboard.generalClipboard`. JavaScript-code in een niet-toepassingssandbox heeft toegang tot het klembord via het gebeurtenisobject dat is verzonden als reactie op een van de kopiëren-en-plakken-gebeurtenissen die zijn verzonden door een element in een HTML-document.

Kopiëren-en-plakken-gebeurtenissen zijn onder andere: `copy`, `cut` en `paste`. Het object dat voor deze gebeurtenissen is verzonden, biedt toegang tot het klembord via de eigenschap `clipboardData`.

Standaardgedrag

Adobe AIR 1.0 of hoger

Als reactie op de kopieeropdracht worden de geselecteerde items standaard door AIR gekopieerd. Deze opdracht kan via een sneltoets of een snelmenu worden gegenereerd. In bewerkbare zones wordt tekst als reactie op de knipopdracht door AIR geknipt, of als reactie op de plakopdracht door AIR geplakt op de cursorpositie of de geselecteerde tekst.

Als u een ander gedrag wenst, kan uw gebeurtenishandler de methode `preventDefault()` van het verzonden gebeurtenisobject oproepen.

Eigenschap clipboardData van gebeurtenisobject gebruiken

Adobe AIR 1.0 of hoger

Met de eigenschap `clipboardData` van het gebeurtenisobject dat als reactie op een van de kopieer- of plakgebeurtenissen is verzonden, kunt u klembordgegevens lezen en schrijven.

Als u naar het klembord wilt schrijven bij het werken met een kopieer- of knipgebeurtenis, gebruikt u de methode `setData()` van het object `clipboardData`, waarbij u de te kopiëren gegevens en het MIME-type opgeeft:

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

Als u toegang wilt tot de gegevens die worden geplakt, gebruikt u de methode `getData()` van het object `clipboardData`, waarbij u het MIME-type van de gegevensindeling opgeeft. De beschikbare indelingen worden weergegeven door de eigenschap `types`.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

De methode `getData()` en de eigenschap `types` zijn alleen toegankelijk in het gebeurtenisobject dat is verzonden door de gebeurtenis `paste`.

Het volgende voorbeeld geeft aan hoe u het standaardgedrag bij kopiëren en plakken in een HTML-pagina kunt wijzigen. De gebeurtenishandler `copy` maakt de gekopieerde tekst cursief en kopieert deze als HTML-tekst naar het klembord. De gebeurtenishandler `cut` kopieert de geselecteerde gegevens naar het klembord en verwijdert deze uit het document. De gebeurtenishandler `paste` voegt de inhoud van het klembord als HTML in en maakt de ingevoegde tekst vet.

```
 <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>
```

Klembordgegevensindelingen

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Klembordindelingen beschreven de gegevens die in een klembordobject zijn geplaatst. Flash Player of AIR zet de standaard gegevensindelingen automatisch om van ActionScript-gegevenstypen in systeemklembordindelingen, en omgekeerd. Bovendien kunnen toepassingsobjecten worden overgebracht binnen en tussen ActionScript-toepassingen met behulp van toepassingsspecifieke indelingen.

Een klembordobject kan dezelfde informatie in meerdere indelingen bevatten. Een klembordobject dat een sprite representeert, kan bijvoorbeeld een referentie-indeling bevatten voor gebruik in dezelfde toepassing, een geserialiseerde indeling voor gebruik door een andere toepassing die wordt uitgevoerd in Flash Player of AIR, een bitmapindeling voor gebruik door een grafische editor en een bestandenlijstindeling, mogelijk met uitgestelde rendering om een PNG-bestand te coderen, voor het kopiëren of slepen van een representatie van de sprite naar het bestandssysteem.

Standaard gegevensindelingen

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

De constanten die de standaard indelingsnamen definiëren, staan in de klasse ClipboardFormats:

Constante	Beschrijving
TEXT_FORMAT	Gegevens met tekstindeling worden omgezet in de ActionScript-klasse String, en omgekeerd.
HTML_FORMAT	Tekst met HTML-markeringen.
RICH_TEXT_FORMAT	RTF-gegevens worden omgezet in de ActionScript-klasse ByteArray, en omgekeerd. De RTF-markeringen worden op geen enkele manier geïnterpreteerd of omgezet.
BITMAP_FORMAT	(Alleen AIR) Gegevens met bitmapindeling worden omgezet in de ActionScript-klasse BitmapData, en omgekeerd.
FILE_LIST_FORMAT	(Alleen AIR) Gegevens met bestandenlijstindeling worden omgezet in een array van ActionScript File-objecten, en omgekeerd.
URL_FORMAT	(Alleen AIR) Gegevens met URL-indeling worden omgezet in de ActionScript-klasse String, en omgekeerd.

Tijdens het kopiëren en plakken van gegevens als reactie op een `copy`-, `cut`-, of `paste`-gebeurtenis in HTML-inhoud van een AIR-toepassing, moeten MIME-typen worden gebruikt in plaats van de ClipboardFormat-tekenreeksen. De volgende MIME-typen zijn geldig voor gegevens:

MIME-type	Beschrijving
Tekst	"text/plain"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Bestandenlijst	"application/x-vnd.adobe.air.file-list"

Opmerking: *RTF-gegevens zijn niet beschikbaar via de eigenschap `clipboardData` van het gebeurtenisobject dat is verzonden als reactie op een `paste`-gebeurtenis in HTML-inhoud.*

Aangepaste gegevensindelingen

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

U kunt toepassingsspecifieke, aangepaste indelingen gebruiken om objecten als verwijzingen of geserialiseerde kopieën over te brengen. Referenties zijn alleen geldig binnen dezelfde toepassing. Geserialiseerde objecten kunnen tussen twee toepassingen worden overgedragen, maar kunnen alleen worden gebruikt met objecten die geldig blijven wanneer deze zijn geserialiseerd en gedeserialiseerd. Objecten kunnen doorgaans worden geserialiseerd als hun eigenschappen van een eenvoudig type zijn of als de objecten serialiseerbaar zijn.

Om een geserialiseerd object aan een Klembordobject toe te voegen, stelt u de parameter *serializable* in op `true` wanneer u de `Clipboard.setData()`-methode aanroept. De naam van de indeling kan die van een standaardindeling zijn, of een willekeurige tekenreeks die door uw toepassing wordt gedefinieerd.

Overdrachtsmodi

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Wanneer een object naar het klembord wordt geschreven met behulp van een aangepaste gegevensindeling, kunnen de objectgegevens van een klembord worden gelezen als een referentie of als een geserialiseerde kopie van het originele object. Er zijn vier overdrachtmodussen die bepalen of objecten worden overgedragen als referenties of als geserialiseerde kopieën:

Overdrachtsmodus	Beschrijving
ClipboardTransferModes.ORIGINAL_ONLY	Er wordt alleen een verwijzing geretourneerd. Als er geen verwijzing beschikbaar is, wordt een nulwaarde geretourneerd.
ClipboardTransferModes.ORIGINAL_PREFERRED	Als er een verwijzing beschikbaar is, wordt deze geretourneerd. Als dat niet het geval is, wordt een geserialiseerde kopie geretourneerd.
ClipboardTransferModes.CLONE_ONLY	Er wordt alleen een geserialiseerde kopie geretourneerd. Als er geen geserialiseerde kopie beschikbaar is, levert dit een nulwaarde op.
ClipboardTransferModes.CLONE_PREFERRED	Als er een geserialiseerde kopie beschikbaar is, wordt deze geretourneerd. Als dat niet het geval is, wordt een verwijzing geretourneerd.

Aangepaste gegevensindelingen lezen en schrijven

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een object naar een klembord schrijft, kunt u elke tekenreeks gebruiken die niet begint met de gereserveerde voorvoegsels `air:` of `flash:` voor de parameter *format*. Gebruik dezelfde tekenreeks als indeling voor het lezen van het object. De volgende voorbeelden illustreren hoe u objecten leest van en schrijft naar het klembord:

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

Om een geserialiseerd object van het klembordobject te extraheren (na verslepen of plakken gebruikt u dezelfde indelingsnaam en de overdrachtmodussen `CLONE_ONLY` of `CLONE_PREFFERED`.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

Er wordt altijd een verwijzing toegevoegd aan het klembordobject. Om de referentie uit het klembordobject te extraheren (na slepen of plakken in plaats van de geserialiseerde kopie gebruikt u de overdrachtmodussen `ORIGINAL_ONLY` of `ORIGINAL_PREFFERED`:

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

Referenties zijn alleen geldig als het Klembordobject van de huidige toepassing komt. Gebruik de overdrachtmodus `ORIGINAL_PREFFERED` om de referentie te openen wanneer deze beschikbaar is en de geserialiseerde kloon wanneer de referentie niet beschikbaar is.

Uitgestelde rendering

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Als het maken van een gegevensindeling de processor zwaar belast, kunt u uitgestelde rendering toepassen door een functie in te stellen die de gegevens op aanvraag levert. De functie wordt alleen aangeroepen als een ontvanger van de actie slepen of kopiëren gegevens opvraagt in de uitgestelde indeling.

De renderfunctie wordt met de methode `setDataHandler()` aan een klembordobject toegevoegd. De functie moet de gegevens in de juiste indeling retourneren. Als u bijvoorbeeld `setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)` hebt opgeroepen, moet de functie `writeText()` een tekenreeks retourneren.

Als een gegevensindeling van hetzelfde type aan een klembordobject wordt toegevoegd met de `setData()`-methode, krijgen die gegevens prioriteit boven de uitgestelde versie (de renderfunctie wordt niet aangeroepen). Als dezelfde klembordgegevens een tweede keer worden benaderd, wordt de renderfunctie mogelijk (opnieuw) opgeroepen.

Opmerking: *In Mac OS X werkt uitgestelde rendering alleen met aangepaste gegevensindelingen. Met standaard gegevensindelingen wordt de renderfunctie onmiddellijk opgeroepen.*

Tekst plakken met behulp van uitgestelde rendering

Flash Player 10 of hoger, Adobe AIR 1.0 of hoger

Het volgende voorbeeld illustreert hoe u uitgestelde rendering toepast.

Wanneer de gebruiker op de knop Copy klikt, wist de toepassing het systeemklembord om te zorgen dat er geen gegevens van vorige klembordbewerkingen meer aanwezig zijn. De `setDataHandler()`-methode stelt vervolgens de `renderData()`-functie in als klembordrenderer.

Wanneer de gebruiker de opdracht Plakken selecteert in het contextmenu van het doelveld, opent de toepassing het klembord en stelt de doeltekst in. Aangezien de tekstgegevensindeling op het klembord is ingesteld met een functie in plaats van een tekenreeks, roept het klembord de functie `renderData()` op. De functie `renderData()` retourneert de tekst in de brontekst, waarna deze aan de doeltekst wordt toegewezen.

Let op: als u brontekst bewerkt voordat u op de knop Paste klikt, worden de bewerkingen opgenomen in de geplakte tekst, zelfs als de bewerking plaatsvindt nadat u op de knop Copy hebt geklikt. De renderfunctie kopieert de brontekst namelijk pas nadat op de knop Paste is geklikt. (Wanneer u uitgestelde rendering in een echte toepassing gebruikt, kunt u het best de brongegevens opslaan of vergrendelen om dit probleem te voorkomen.)

Flash-voorbeeld

```
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;
            }
        }
    }
}
```

Flex-voorbeeld

```
<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>
```

Hoofdstuk 34: Versnellingsmeterinvoer

Flash Player 10.1 of hoger, Adobe AIR 2 of hoger

De Accelerometer-klasse verstuurt gebeurtenissen op basis van de activiteit die door de bewegingssensor van het apparaat wordt gedetecteerd. Deze gegevens vertegenwoordigen de locatie of beweging van het apparaat langs een driedimensionale as. Wanneer het apparaat beweegt, detecteert de sensor deze beweging en worden de versnellingscoördinaten van het apparaat geretourneerd. De Accelerometer-klasse biedt methoden om te bepalen of de versnellingsmeter wordt ondersteund en om de snelheid in te stellen waarmee versnellingsgebeurtenissen worden verzonden.

De assen van de versnellingsmeter worden genormaliseerd ten opzichte van de weergave-oriëntatie, niet ten opzichte van de fysieke oriëntatie van het apparaat. Wanneer de weergaveoriëntatie van het apparaat wordt veranderd, veranderen de assen van de versnellingsmeter ook. De y-as is dus altijd min of meer verticaal wanneer de gebruiker de telefoon gewoon rechtop houdt, ongeacht de manier waarop de telefoon is gedraaid. Als automatische oriëntatie uitgeschakeld is, bijvoorbeeld wanneer SWF-inhoud op volledig scherm in een browser wordt weergegeven, worden de assen van de versnellingsmeter niet aangepast wanneer het apparaat wordt gedraaid.

Meer Help-onderwerpen

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

Controleren op ondersteuning voor de versnellingsmeter

Met de eigenschap `Accelorometer.isSupported` kunt u testen of deze functie kan worden gebruikt in de runtimeomgeving:

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

De Accelerometer-klasse en de bijbehorende leden zijn toegankelijk voor de runtimeversies die voor elke API-invoer worden vermeld. Of deze functie ook werkelijk beschikbaar is, wordt bepaald door de actuele omgeving tijdens runtime. U kunt bijvoorbeeld code compileren met de eigenschappen van de Accelerometer-klasse voor Flash Player 10.1, maar u moet de eigenschap `Accelerometer.isSupported` gebruiken om te testen op de beschikbaarheid van de Accelerometer-functie op het gebruikersapparaat. Als de waarde van `Accelerometer.isSupported` gelijk is aan `true` tijdens runtime, is er ondersteuning voor de Accelerometer-functie.

Wijzigingen in de versnellingsmeter vaststellen

Als u de versnellingsmetersensor wilt gebruiken, moet u een Accelerometer-object instantiëren en registeren voor de `update`-gebeurtenissen die door het object worden verzonden. De `update`-gebeurtenis is een Accelerometer -object. De gebeurtenis heeft vier eigenschappen, allemaal getallen.

- `accelerationX`: versnelling langs de x-as, gemeten in g. De x-as loopt van de linkerkant van het apparaat naar de rechterkant als het apparaat rechtop staat. (Het apparaat staat rechtop als de bovenkant van het apparaat naar boven is gericht.) De versnelling is positief als het apparaat naar rechts beweegt.
- `accelerationY`: versnelling langs de y-as, gemeten in g. De y-as loopt van de onderkant van het apparaat naar de bovenkant als het apparaat rechtop staat. (Het apparaat staat rechtop als de bovenkant van het apparaat naar boven is gericht.) De versnelling is positief als het apparaat naar boven beweegt.
- `accelerationZ`: versnelling langs de z-as, gemeten in g. De z-as staat loodrecht op de voorzijde van het apparaat. De versnelling is positief als u het apparaat verplaatst zodat de voorkant van het apparaat naar boven wijst. De versnelling is negatief als de voorkant van het apparaat naar beneden wijst.
- `timestamp`: het aantal milliseconden na het initialiseren van het runtimeprogramma waarop de gebeurtenis plaatsvindt.

1 g is de standaardversnelling van de zwaartekracht (ongeveer 9,8 m/sec^2).

In dit voorbeeld worden de gegevens over de versnelling in een tekstveld weergegeven:

```
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()
}
```

Als u dit voorbeeld wilt gebruiken, moet u het tekstveld`accTextField` maken en dit toevoegen aan de weergavelijst voordat u deze code gebruikt.

U kunt het gewenste tijdsinterval voor versnellingsmetergebeurtenissen aanpassen door de methode `setRequestedUpdateInterval()` van het Accelerometer-object aan te roepen. Deze methode neemt één parameter, `interval`, die staat voor het gevraagde update-interval in milliseconden:

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

De werkelijke tijd tussen versnellingsmeterupdates kan groter of kleiner zijn dan deze waarde. Elke wijziging in het update-interval beïnvloedt alle geregistreerde listeners. Als u de methode `setRequestedUpdateInterval()` niet aanroept, ontvangt de toepassing updates op basis van het standaardinterval voor het apparaat.

De gegevens van de versnellingsmeter zijn niet 100% nauwkeurig. Door het gemiddelde van recente bewegingsgegevens toe te passen kunt u de gegevens betrouwbaarder maken. In het volgende voorbeeld worden recente lezingen van de versnellingsmeter ingecalculeerd bij de huidige lezing om tot een afgerond resultaat te komen:

```
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));
}
```

Dit gemiddelde is alleen gewenst bij kleine update-intervallen van de versnellingsmeter.

Hoofdstuk 35: Slepen en neerzetten in AIR

Adobe AIR 1.0 of hoger

Gebruik de klassen in de Adobe® AIR™ API voor slepen en neerzetten als ondersteuning voor slepen en neerzetten in de gebruikersinterface. Een *beweging* is in deze context een actie van de gebruiker die wordt overgebracht door het besturingssysteem en uw toepassing, met de bedoeling om gegevens de kopiëren, te verplaatsen of te koppelen. Een *uitsleep*beweging treedt op wanneer de gebruiker een object uit een component of een toepassing sleept. Een *insleep*beweging treedt op wanneer de gebruiker een object naar een component of een toepassing sleept.

Via de API voor slepen en neerzetten kan een gebruiker gegevens slepen tussen toepassingen en tussen componenten in een toepassing. Onder andere de volgende overdrachtsindelingen worden ondersteund:

- Bitmaps
- Bestanden
- Tekst met HTML-opmaak
- Tekst
- RTF-gegevens
- URL's
- Promises, bestand
- Geserialiseerde objecten
- Objectverwijzingen (alleen geldig binnen de brontoepassing)

Basisprincipes van slepen en neerzetten in AIR

Adobe AIR 1.0 of hoger

Lees voor een snelle uitleg van en codevoorbeelden van het gebruik van slepen en neerzetten in een AIR-toepassing, de volgende artikelen voor snel starten in de Adobe Developer Connection:

- Ondersteuning voor slepen en neerzetten en voor kopiëren en plakken (Flex)
- Ondersteuning voor slepen en neerzetten en voor kopiëren en plakken (Flash)

De API voor slepen en neerzetten bevat de volgende klassen.

Pakket	Klassen
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise Constanten die worden gebruikt met de API voor slepen en neerzetten, zijn gedefinieerd in de volgende klassen: • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

Stadia van bewegingen voor slepen en neerzetten

De beweging voor slepen en neerzetten bestaat uit drie stadia:

Initiëren *Een gebruiker start het slepen en neerzetten door te slepen vanuit een component of een item in een component terwijl de muisknop is ingedrukt.* De component waaruit het gesleepte item afkomstig is, wordt gewoonlijk de sleepinitiator genoemd en deze verzendt de gebeurtenissen `nativeDragStart` en `nativeDragComplete`. Een Adobe AIR-toepassing start een sleepactie door de `NativeDragManager.doDrag()`-methode aan te roepen na een `mouseDown`- of `mouseMove`-gebeurtenis.

Als de sleepbewerking is gestart van buiten een AIR-toepassing, is er geen initiator-object om een `nativeDragStart`- of `nativeDragComplete`-gebeurtenis te verzenden.

Slepen *Terwijl de muisknop is ingedrukt, verplaatst de gebruiker de muiscursor naar een andere component of toepassing, of naar het bureaublad.* Zolang het slepen duurt, verzendt het initiatorobject `nativeDragUpdate`-gebeurtenissen. (Deze gebeurtenis wordt echter voor Linux niet in AIR verzonden. Wanneer de gebruiker de muis over een mogelijk neerzetdoel beweegt, verzendt het neerzetdoel de gebeurtenis `nativeDragEnter`. De gebeurtenishandler kan het gebeurtenisobject inspecteren om te bepalen of de gesleepte gegevens beschikbaar zijn in een indeling die het doel accepteert, en als dat het geval is, de gebruiker de gegevens daarop laten neerzetten door de methode `NativeDragManager.acceptDragDrop()` op te roepen.

Zolang de sleepbeweging over een interactief object plaatsvindt, verzendt dat object `nativeDragOver`-gebeurtenissen. Wanneer de sleepbeweging het interactieve object verlaat, verzendt het object de gebeurtenis `nativeDragExit`.

Neerzetten *De gebruiker laat de muisknop los op een geschikt neerzetdoel.* Als het doel een AIR-toepasing of -component is, verzendt het doelobject een `nativeDragDrop`-gebeurtenis. De gebeurtenishandler kan toegang krijgen tot de gegevens die door het gebeurtenisobject worden overgedragen. Als het doel zich buiten AIR bevindt, wordt het neerzetten afgehandeld door het besturingssysteem of een andere toepassing. In beide gevallen verzendt het object dat de beweging initieert de gebeurtenis `nativeDragComplete` (als het slepen in AIR is gestart).

Met de klasse NativeDragManager worden zowel insleep- als uitsleepbewegingen beheerd. Alle leden van de klasse NativeDragManager zijn statisch; maak geen instantie van deze klasse.

Clipboard-objecten
Gegevens die in of uit een toepassing of component worden gesleept, bevinden zich in een Clipboard-object. Eén Clipboard-object kan verschillende weergaven van dezelfde gegevens beschikbaar maken om de kans te vergroten dat een andere toepassing de gegevens kan begrijpen en gebruiken. Een afbeelding kan bijvoorbeeld worden opgenomen als afbeeldingsgegevens, een geserialiseerd bitmapobject en een bestand. Het omzetten van de gegevens naar een bepaalde indeling kan worden uitgevoerd door een renderfunctie die pas wordt opgeroepen als de gegevens zijn gelezen.

Wanneer een sleepbeweging is gestart, is het Clipboard-object alleen toegankelijk vanuit een gebeurtenishandler voor de gebeurtenissen `nativeDragEnter`, `nativeDragOver` en `nativeDragDrop`. Nadat de sleepbeweging is beëindigd, kan het Clipboard-object niet worden gelezen of opnieuw worden gebruikt.

Een toepassingsobject kan worden overgedragen als een verwijzing en als een geserialiseerd object. Verwijzingen zijn alleen geldig binnen de brontoepassing. Geserialiseerde objecten kunnen worden overgedragen tussen AIR-toepassingen, maar kunnen alleen worden gebruikt met objecten die geldig blijven wanneer ze zijn geserialiseerd en gedeserialiseerd. Objecten die zijn geserialiseerd, worden geconverteerd naar de AMF3-indeling (Action Message Format voor ActionScript 3), een indeling voor gegevensoverdracht op basis van tekenreeksen.

Werken met het Flex-framework
In de meeste gevallen is het beter om de API voor slepen en neerzetten van Adobe® Flex™ te gebruiken bij het bouwen van Flex-toepassingen. Het Flex-framework beschikt over een equivalente functieset wanneer een Flex-toepassing wordt uitgevoerd in AIR (het maakt intern gebruik van de NativeDragManager van AIR). Flex beschikt ook over een beperkte functieset wanneer een toepassing of component wordt uitgevoerd in de browseromgeving, die meer beperkingen heeft. AIR-klassen kunnen niet worden gebruikt in componenten of toepassingen die buiten de runtimeomgeving van AIR worden uitgevoerd.

Ondersteuning voor de uitsleepbeweging

Adobe AIR 1.0 of hoger

Voor ondersteuning van de uitsleepbeweging moet u een Clipboard-object maken als reactie op de gebeurtenis `mouseDown` en deze naar de methode `NativeDragManager.doDrag()` verzenden. Uw toepassing kan vervolgens naar de gebeurtenis `nativeDragComplete` van het initiatorobject luisteren om te bepalen welke actie moet worden uitgevoerd wanneer de gebruiker de beweging voltooit of annuleert.

Gegevens voorbereiden voor overdracht

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Om gegevens of een object voor te bereiden op slepen, maakt u een Clipboard-object en voegt u de gegevens die moeten worden overgebracht toe in een of meer indelingen. U kunt de standaardgegevensindelingen gebruiken voor het doorgeven van gegevens die automatisch kunnen worden omgezet in eigen klembordindelingen en toepassingsspecifieke indelingen gebruiken voor het doorgeven van objecten.

Als het een grote belasting voor de processor is om de gegevens die moeten worden overgebracht, te converteren naar een bepaalde indeling, kunt u de naam opgeven van een handlerfunctie die de conversie uitvoert. Deze functie wordt alleen opgeroepen als de ontvangende component of toepassing de bijbehorende indeling leest.

Zie “Klembordgegevensindelingen” op pagina 632 voor meer informatie over de indelingen voor klembordgegevens.

In het volgende voorbeeld wordt een Clipboard-object gemaakt dat een bitmap in diverse indelingen bevat: een bitmapobject, een native bitmapindeling en een bestandenlijstindeling die het bestand bevat waaruit de bitmap oorspronkelijk is geladen:

```
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;
}
```

Uitsleepbewerkingen starten

Adobe AIR 1.0 of hoger

Om een sleepbewerking te starten, roept u de methode `NativeDragManager.doDrag()` op als reactie op het indrukken van de muis. De methode `doDrag()` is een statische methode met de volgende parameters:

Parameter	Beschrijving
initiator	Het object van waaruit wordt gesleept en dat de gebeurtenissen `dragStart` en `dragComplete` verzendt. De initiator moet een interactief object zijn.
klembord	Het Clipboard-object dat de gegevens bevat die moeten worden overgedragen. Naar het Clipboard-object wordt verwezen in de NativeDragEvent-objecten die worden verzonden tijdens het slepen en neerzetten.
dragImage	(Optioneel) Een BitmapData-object dat tijdens het slepen wordt weergegeven. De afbeelding kan een `alfa`-waarde opgeven. (Opmerking: Microsoft Windows past altijd een vaste alfa-vervaging toe bij het verslepen van afbeeldingen.)
offset	(Optioneel) Een Point-object waarmee de afstand van de sleepafbeelding vanaf de hotspot van de muis wordt opgegeven. Gebruik negatieve coördinaten om de sleepafbeelding naar boven en naar links ten opzichte van de muiscursor te verplaatsen. Als er geen afstand is opgegeven, wordt de linkerbovenhoek van de sleepafbeelding op de hotspot van de muis geplaatst.
actionsAllowed	(Optioneel) Een NativeDragOptions-object waarmee de acties (kopiëren, verplaatsen of koppelen) worden opgegeven die geldig zijn voor de sleepbewerking. Als er geen argument is opgegeven, zijn alle acties toegestaan. In NativeDragEvent-objecten wordt naar het DragOptions-object verwezen om het mogelijk te maken dat een sleepdoel controleert of de toegestane acties compatibel zijn met de functie van de doelcomponent. Een “prullenbak”-component accepteert bijvoorbeeld alleen sleepbewegingen voor een verplaatsingsactie.

In het volgende voorbeeld ziet u hoe u een sleepbewerking start voor een bitmapobject dat uit een bestand is geladen. In het voorbeeld wordt een afbeelding geladen en wordt de sleepbewerking gestart bij de gebeurtenis `mouseDown`.

```
 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);
    }
}
}
```

Uitsleepoverdrachten voltooien

Adobe AIR 1.0 of hoger

Wanneer een gebruiker het gesleepte item neerzet door de muisknop los te laten, verzendt het initiatorobject de gebeurtenis `nativeDragComplete`. U kunt de eigenschap `dropAction` van het gebeurtenisobject controleren en vervolgens de juiste actie uitvoeren. Als de actie bijvoorbeeld `NativeDragAction.MOVE`, is, kunt u het bronitem op de bronlocatie verwijderen. De gebruiker kan een sleepbeweging annuleren door de muisknop los te laten terwijl de cursor zich buiten een geschikt neerzetdoel bevindt. De sleepbeheerder stelt de eigenschap `dropAction` voor een geannuleerde beweging in op `NativeDragAction.NONE`.

Ondersteuning voor de insleepbeweging

Adobe AIR 1.0 of hoger

Voor ondersteuning van de insleepbeweging moet uw toepassing (gewoonlijk een visuele component van uw toepassing) reageren op de gebeurtenis `nativeDragEnter` of `nativeDragOver`.

Stappen waaruit een neerzetbewerking gewoonlijk bestaat

Adobe AIR 1.0 of hoger

Een neerzetbewerking bestaat gewoonlijk uit de volgende reeks gebeurtenissen:

1. De gebruiker sleept een Clipboard-object over een component.
2. De component verzendt de gebeurtenis `nativeDragEnter`.
3. De gebeurtenishandler `nativeDragEnter` controleert de beschikbare gegevensindelingen en toegestane acties van het gebeurtenisobject. Als de component het neerzetten kan verwerken, wordt `NativeDragManager.acceptDragDrop()` opgeroepen.

4 NativeDragManager wijzigt de muiscursor om aan te geven dat het object kan worden neergezet.

5 De gebruiker zet het object neer op de component.

6 De ontvangende component verzendt de gebeurtenis `nativeDragDrop`.

7 De ontvangende component leest de gegevens in de gewenste indeling uit het Clipboard-object in het gebeurtenisobject.

8 Als de sleepbeweging is gestart in een AIR-toepassing, verzendt het initiërende interactieve object de gebeurtenis `nativeDragComplete`. Als de beweging buiten AIR is gestart, wordt geen feedback verzonden.

Insleepbewegingen bevestigen

Adobe AIR 1.0 of hoger

Wanneer een gebruiker een klemborditem binnen de grenzen van een visuele component sleept, verzendt de component de gebeurtenissen `nativeDragEnter` en `nativeDragOver`. Om te bepalen of de component het klemborditem kan accepteren, kunnen de handlers voor deze gebeurtenissen de eigenschappen `clipboard` en `allowedActions` van het gebeurtenisobject controleren. De gebeurtenishandler moet de methode `NativeDragManager.acceptDragDrop()` met een verwijzing naar de ontvangende component oproepen om aan te geven dat de component het neerzetten kan accepteren. Als meerdere geregistreerde gebeurtenislisteners de methode `acceptDragDrop()` oproepen, heeft de laatste handler in de lijst de hoogste prioriteit. De oproep `acceptDragDrop()` blijft geldig tot de muis buiten de grenzen van het accepterende object komt, waardoor de gebeurtenis `nativeDragExit` wordt geactiveerd.

Als er meerdere acties zijn toegestaan in de parameter `allowedActions` die aan `doDrag()` is doorgegeven, kan de gebruiker een aanpassingstoets ingedrukt houden om aan te geven welke actie moet worden uitgevoerd. De sleepbeheerder wijzigt de cursorafbeelding om de gebruiker te laten weten welke actie wordt uitgevoerd als het neerzetten wordt voltooid. De bedoelde actie wordt gerapporteerd aan de eigenschap `dropAction` van het NativeDragEvent-object. De ingestelde actie voor een sleepbeweging is slechts een voorstel. De componenten die bij de overdracht zijn betrokken, moeten het juiste gedrag implementeren. Het gesleepte item moet bijvoorbeeld door de sleepinitiator worden verwijderd en door het neerzetdoel worden toegevoegd om een verplaatsingsactie te voltooien.

De neerzetactie kan door het sleepdoel tot een van de drie mogelijke acties worden beperkt door de eigenschap `dropAction` van de klasse NativeDragManager in te stellen. Als een gebruiker een andere actie wil kiezen met het toetsenbord, wordt de cursor voor een *niet-beschikbare* actie weergegeven. Stel de eigenschap `dropAction` in de handlers voor de gebeurtenissen `nativeDragEnter` en `nativeDragOver` in.

In het volgende voorbeeld ziet u een gebeurtenishandler voor de gebeurtenis `nativeDragEnter` of `nativeDragOver`. Deze handler accepteert alleen een insleepbeweging als het klembord dat wordt gesleept tekst bevat.

```
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
    }
}
```

Het neerzetten voltooien

Adobe AIR 1.0 of hoger

Wanneer de gebruiker een gesleept item neerzet op een interactief object dat de beweging heeft geaccepteerd, verzendt het interactieve object de gebeurtenis `nativeDragDrop`. De handler voor deze gebeurtenis kan de gegevens ophalen uit de eigenschap `clipboard` van het gebeurtenisobject.

Wanneer het klembord een toepassingsspecifieke indeling bevat, bepaalt de parameter `transferMode` die aan de methode `getData()` van het Clipboard-object is doorgegeven, of de sleepbeheerder een verwijzing of een geserialiseerde versie van het object retourneert.

In het volgende voorbeeld ziet u een gebeurtenishandler voor de gebeurtenis `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));
}
```

Wanneer de gebeurtenishandler is afgesloten, is het Clipboard-object niet meer geldig. Bij elke poging om toegang tot het object of de gegevens te krijgen, wordt een fout gegenereerd.

De visuele weergave van een component bijwerken

Adobe AIR 1.0 of hoger

De visuele weergave van een component kan worden bijgewerkt op basis van de NativeDragEvent-gebeurtenissen. In de volgende tabel wordt beschreven welke typen wijzigingen een component gewoonlijk aanbrengt als reactie op de verschillende gebeurtenissen:

Gebeurtenis	Beschrijving
nativeDragStart	Het initiërende interactieve object kan met de gebeurtenis `nativeDragStart` visueel aangeven dat de sleepbeweging afkomstig is van dat interactieve object.
nativeDragUpdate	Het initiërende interactieve object kan de gebeurtenis nativeDragUpdate gebruiken om de status bij te werken tijdens de beweging. (Deze gebeurtenis bestaat niet in AIR voor Linux.)
nativeDragEnter	Een mogelijk ontvangend interactief object kan deze gebeurtenis gebruiken om de focus te nemen of visueel aangeven dat het neerzetten wel of niet wordt geaccepteerd.
nativeDragOver	Een mogelijk ontvangend interactief object kan deze gebeurtenis gebruiken om te reageren op de beweging van de muis binnen het interactieve object, bijvoorbeeld wanneer de muis een dynamische regio van een complexe component, zoals een stratenplan, bereikt.
nativeDragExit	Een mogelijk ontvangend interactief object kan deze gebeurtenis gebruiken om de status te herstellen wanneer een sleepbeweging buiten de grenzen komt.
nativeDragComplete	Het initiërende interactieve object kan deze gebeurtenis gebruiken om het bijbehorende gegevensmodel bij te werken, zoals het verwijderen van een item uit een lijst, en om de visuele status te herstellen.

Muisposities bijhouden tijdens een sleepbeweging

Adobe AIR 1.0 of hoger

Terwijl de muis over een component wordt gesleept, verzendt die component `nativeDragOver`-gebeurtenissen. Deze gebeurtenissen worden elke paar milliseconden verzonden en ook wanneer de muis beweegt. Het gebeurtenisobject `nativeDragOver` kan worden gebruikt om de positie van de muis in de component te bepalen. Toegang tot de muispositie kan nuttig zijn in situaties waarin de ontvangende component complex is maar niet uit subcomponenten bestaat. Als uw toepassing bijvoorbeeld een bitmap met een stratenplan weergeeft en u zones op de kaart wilt markeren wanneer de gebruiker daar gegevens heen sleept, kunt u met de muiscoördinaten in de gebeurtenis `nativeDragOver` de muispositie op de kaart bijhouden.

Slepen en neerzetten in HTML

Adobe AIR 1.0 of hoger

Als u gegevens in en uit een HTML-toepassing wilt slepen (of in en uit de HTML die wordt weergegeven in een HTMLLoader), kunt u gebeurtenissen voor het slepen en neerzetten van HTML gebruiken. Met de API voor het slepen en neerzetten van HTML kunt u gegevens slepen naar en vanuit DOM-elementen in de HTML-inhoud.

Opmerking: *U kunt ook de API's NativeDragEvent en NativeDragManager van AIR gebruiken door te luisteren naar gebeurtenissen van het HTMLLoader-object dat de HTML-inhoud bevat. De API voor HTML is echter beter geïntegreerd met het HTML DOM en geeft u meer controle over het standaardgedrag.*

Standaardgedrag bij slepen en neerzetten

Adobe AIR 1.0 of hoger

De HTML-omgeving biedt standaardgedrag voor bewegingen die betrekking hebben op het slepen en neerzetten van tekst, afbeeldingen en URL's. Met het standaardgedrag kunt u deze typen gegevens altijd uit een element slepen. U kunt echter alleen tekst naar een element slepen en alleen naar elementen in een bewerkbaar gebied op een pagina. Wanneer u tekst sleept tussen of in bewerkbare gebieden op een pagina, wordt de tekst standaard verplaatst. Wanneer u tekst vanuit een niet-bewerkbaar gebied of van buiten de toepassing naar een bewerkbaar gebied sleept, wordt de tekst standaard gekopieerd.

U kunt het standaardgedrag overschrijven door de gebeurtenissen voor slepen en neerzetten zelf af te handelen. Als u het standaardgedrag wilt annuleren, moet u de methode `preventDefault()` oproepen van de objecten die worden verzonden voor gebeurtenissen voor slepen en neerzetten. Vervolgens kunt u gegevens in het neerzetdoel invoegen en gegevens uit het sleepdoel verwijderen om de gekozen actie uit te voeren.

Standaard kan de gebruiker elke tekst selecteren en slepen, en afbeeldingen en koppelingen slepen. Met de CSS-eigenschap `-webkit-user-select` van de WebKit kunt u bepalen hoe elk HTML-element kan worden geselecteerd. Als u bijvoorbeeld `-webkit-user-select` op `none` instelt, kan de inhoud van het element niet worden geselecteerd en dus ook niet worden gesleept. U kunt ook de CSS-eigenschap `-webkit-user-drag` gebruiken om te bepalen of een element in zijn geheel kan worden gesleept. De inhoud van het element wordt echter afzonderlijk behandeld. De gebruiker kan ook een geselecteerd gedeelte van de tekst slepen. Zie "CSS in AIR" op pagina 1034 voor meer informatie.

Gebeurtenissen voor slepen en neerzetten in HTML

Adobe AIR 1.0 of hoger

Het initiatorelement van waaruit een sleepbeweging afkomstig is, verzendt de volgende gebeurtenissen:

Gebeurtenis	Beschrijving
dragstart	Wordt verzonden wanneer de gebruiker de sleepbeweging start. De handler voor deze gebeurtenis kan indien nodig het slepen voorkomen door de methode preventDefault() van het gebeurtenisobject op te roepen. Stel de eigenschap effectAllowed in om te bepalen of de gesleepte gegevens kunnen worden gekopieerd, gekoppeld of verplaatst. Geselecteerde tekst, afbeeldingen en koppelingen worden standaard op het klembord geplaatst, maar u kunt verschillende gegevens voor de sleepbeweging instellen met de eigenschap dataTransfer van het gebeurtenisobject.
drag	Wordt continu verzonden tijdens de sleepbeweging.
dragend	Wordt verzonden wanneer de gebruiker de muisknop loslaat om de sleepbeweging te beëindigen.

Een sleepdoel verzendt de volgende gebeurtenissen:

Gebeurtenis	Beschrijving
dragover	Wordt continu verzonden zolang de sleepbeweging plaatsvindt binnen de grenzen van het element. De handler voor deze gebeurtenis moet de eigenschap dataTransfer.dropEffect instellen om aan te geven of de neergezette gegevens worden gekopieerd, verplaatst of gekoppeld wanneer de gebruiker de muisknop loslaat.
dragenter	Wordt verzonden wanneer de sleepbeweging binnen de grenzen van het element komt. Als u eigenschappen van een dataTransfer-object wijzigt in een handler voor de dragenter-gebeurtenis, worden die wijzigingen overschreven door de volgende dragover-gebeurtenis. Er is echter een korte vertraging tussen een dragenter-gebeurtenis en de eerste dragover-gebeurtenis, waardoor de cursor kan gaan knipperen als andere eigenschappen zijn ingesteld. In veel gevallen kunt u dezelfde gebeurtenishandler gebruiken voor beide gebeurtenissen.
dragleave	Wordt verzonden wanneer de sleepbeweging buiten de grenzen van het element komt.
drop	Wordt verzonden wanneer de gebruiker de gegevens op het element neerzet. De gesleepte gegevens zijn alleen toegankelijk in de handler voor deze gebeurtenis.

Het gebeurtenisobject dat wordt verzonden als reactie op deze gebeurtenissen, is vergelijkbaar met een muisgebeurtenis. Met eigenschappen van een muisgebeurtenis, zoals (`clientX`, `clientY`) en (`screenX`, `screenY`), kunt u de positie van de muis bepalen.

De belangrijkste eigenschap van een sleepgebeurtenisobject is `dataTransfer`, dat de gegevens bevat die worden gesleept. Het `dataTransfer`-object heeft de volgende eigenschappen en methoden:

Eigenschap of methode	Beschrijving
effectAllowed	Het effect dat is toegestaan door de bron van waaruit wordt gesleept. Gewoonlijk wordt deze waarde ingesteld door de handler voor de gebeurtenis dragstart. Zie "Sleepeffecten in HTML" op pagina 653.
dropEffect	Het effect dat is gekozen door het doel of door de gebruiker. Als u `dropEffect` instelt in een gebeurtenishandler voor `dragover` of `dragenter`, wordt de muiscursor door AIR bijgewerkt en geeft deze het effect aan dat optreedt als de gebruiker de muis loslaat. Als de ingestelde waarde voor `dropEffect` niet overeenkomt met een van de toegestane effecten, is neerzetten niet toegestaan en wordt de cursor voor een *niet-beschikbare* actie weergegeven. Als u geen waarde voor `dropEffect` hebt ingesteld als reactie op de laatste `dragover`- of `dragenter`-gebeurtenis, kan de gebruiker uit de toegestane effecten kiezen met de standaard aanpassingstoetsen van het besturingssysteem. Het uiteindelijke effect wordt gerapporteerd door de eigenschap `dropEffect` van het object dat wordt verzonden voor `dragend`. Als de gebruiker het neerzetten annuleert door de muisknop los te laten buiten een geschikt doel, wordt `dropEffect` ingesteld op `none`.
types	Een array met de MIME-tekenreeksen voor elke gegevensindeling die aanwezig is in het object `dataTransfer`.
getData(mimeType)	Hiermee worden de gegevens opgehaald in de indeling die is opgegeven met de parameter `mimeType`. De methode `getData()` kan alleen worden opgeroepen als reactie op de gebeurtenis `drop`.
setData(mimeType)	Hiermee worden gegevens aan `dataTransfer` toegevoegd in de indeling die is opgegeven met de parameter mimeType. U kunt gegevens in meerdere indelingen toevoegen door `setData()` op te roepen voor elk MIME-type. Alle gegevens die door het standaardgedrag voor neerzetten in het object `dataTransfer` worden geplaatst, worden gewist. De methode `setData()` kan alleen worden opgeroepen als reactie op de gebeurtenis `dragstart`.
clearData(mimeType)	Hiermee worden alle gegevens gewist in de indeling die is opgegeven met de parameter `mimeType`.
setDragImage(image, offsetX, offsetY)	Hiermee stelt u een aangepaste sleepafbeelding in. De `setDragImage()`-methode kan alleen worden aangeroepen als reactie op een dragstart-gebeurtenis en alleen wanneer een volledig HTML-element wordt versleept door zijn CSS-stijl `-webkit-user-drag` op `element` in te stellen. De parameter `image` kan een JavaScript-element of Image-object zijn.

MIME-typen voor het slepen en neerzetten van HTML

Adobe AIR 1.0 of hoger

U kunt onder andere de volgende MIME-typen gebruiken met het object `dataTransfer` van een gebeurtenis voor het slepen en neerzetten van HTML:

Gegevensindeling	MIME-type
Tekst	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Bestandenlijst	"application/x-vnd.adobe.air.file-list"

U kunt ook andere MIME-tekenreeksen gebruiken, zoals tekenreeksen die door de toepassing zijn gedefinieerd. Het is echter mogelijk dat andere toepassingen de overgedragen gegevens niet herkennen of kunnen gebruiken. Het is uw verantwoordelijkheid om gegevens aan het object `dataTransfer` toe te voegen in de verwachte indeling.

Belangrijk: *Alleen code die in de toepassingssandbox wordt uitgevoerd, heeft toegang tot neergezette bestanden. Als een eigenschap van een File-object wordt gelezen of ingesteld in een externe sandbox, wordt een beveiligingsfout gegenereerd.* *Zie "Bestanden neerzetten in niet-toepassingssandboxen in HMTL" op pagina 657 voor meer informatie.*

Sleepeffecten in HTML

Adobe AIR 1.0 of hoger

De initiator van de sleepbeweging kan de toegestane sleepeffecten beperken door de eigenschap `dataTransfer.effectAllowed` in te stellen in de handler voor de gebeurtenis `dragstart`. De volgende tekenreeken kunnen worden gebruikt:

Tekenreekswaarde	Beschrijving
"none"	Er zijn geen sleepbewerkingen toegestaan.
"copy"	De gegevens worden naar het doel gekopieerd terwijl de oorspronkelijke gegevens op hun plaats blijven staan.
"link"	De gegevens worden met het neerzetdoel gedeeld via een koppeling met de oorspronkelijke gegevens.
"move"	De gegevens worden naar het doel gekopieerd en van de oorspronkelijke locatie verwijderd.
"copyLink"	De gegevens kunnen worden gekopieerd of gekoppeld.
"copyMove"	De gegevens kunnen worden gekopieerd of verplaatst.
"linkMove"	De gegevens kunnen worden gekoppeld of verplaatst.
"all"	De gegevens kunnen worden gekopieerd, verplaatst of gekoppeld. *All* is het standaardeffect wanneer u het standaardgedrag uitschakelt.

Het doel van de sleepbeweging kan de eigenschap `dataTransfer.dropEffect` instellen om aan te geven welke actie wordt uitgevoerd als de gebruiker de gegevens heeft neergezet. Als het neerzeteffect een van de toegestane acties is, wordt de juiste cursor voor kopiëren, verplaatsen of koppelen weergegeven. Als dat niet zo is, wordt de cursor voor een *niet-beschikbare* actie weergegeven. Als door het doel geen neerzeteffect is ingesteld, kan de gebruiker een toegestane actie kiezen met de aanpassingstoetsen.

Stel de waarde van `dropEffect` in de handlers voor de gebeurtenissen `dragover` en `dragenter` in:

```
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";
}
```

Opmerking: *Hoewel u de eigenschap `dropEffect` altijd moet instellen in de handler voor `dragenter`, moet u niet vergeten dat de volgende `dragover`-gebeurtenis de standaardwaarde van de eigenschap herstelt. Stel `dropEffect` in als reactie op beide gebeurtenissen.*

Gegevens uit een HTML-element slepen

Adobe AIR 1.0 of hoger

Bij het standaardgedrag kan de meeste inhoud van een HTML-pagina via slepen worden gekopieerd. Met de CSS-eigenschappen `-webkit-user-select` en `-webkit-user-drag` kunt u bepalen welke inhoud mag worden gesleept.

Overschrijf het standaardgedrag voor uitslepen in de handler voor de gebeurtenis `dragstart`. Roep de methode `setData()` van de eigenschap `dataTransfer` van het gebeurtenisobject op om uw eigen gegevens in de sleepbeweging te plaatsen.

Stel de eigenschap `dataTransfer.effectAllowed` van het gebeurtenisobject dat voor de gebeurtenis `dragstart` wordt verzonden in om aan te geven welke sleepeffecten een bronobject ondersteunt wanneer u geen gebruik maakt van het standaardgedrag. U kunt elke combinatie van effecten kiezen. Als een bronelement bijvoorbeeld de effecten *kopiëren* en *koppelen* ondersteunt, stelt u de eigenschap in op `"copyLink"`.

Gesleepte gegevens instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voeg de gegevens voor de sleepbeweging met de eigenschap `dataTransfer` toe in de handler voor de gebeurtenis `dragstart`. Gebruik de methode `dataTransfer.setData()` om gegevens op het klembord te plaatsen en geef daarbij het MIME-type en de gegevens die u wilt overdragen door.

Als uw toepassing bijvoorbeeld een afbeeldingselement met de id *imageOfGeorge* bevat, kunt u de volgende gebeurtenishandler voor dragstart gebruiken. In dit voorbeeld ziet u hoe weergaven van een afbeelding van George in verschillende gegevensindelingen worden toegevoegd, waardoor de kans groter wordt dat andere toepassingen de gesleepte gegevens kunnen gebruiken.

```
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));
}
```

Opmerking: *Wanneer u de methode `setData()` van het object `dataTransfer` oproept, worden geen gegevens toegevoegd als het standaardgedrag voor slepen en neerzetten wordt gebruikt.*

Gegevens naar een HTML-element slepen

Adobe AIR 1.0 of hoger

Bij het standaardgedrag mag alleen tekst worden gesleept naar bewerkbare gebieden van de pagina. U kunt opgeven dat een element en de onderliggende elementen bewerkbaar mogen worden gemaakt, door het kenmerk `contenteditable` op te nemen in de openingscode van het element. U kunt ook een heel document bewerkbaar maken door de eigenschap `designMode` van het documentobject in te stellen op `"on"`.

U kunt alternatief insleepgedrag op een pagina mogelijk maken door het verwerken van de gebeurtenissen `dragenter`, `dragover` en `drop` voor alle elementen die gesleepte gegevens kunnen accepteren.

Inslepen mogelijk maken

Adobe AIR 1.0 of hoger

Als u de insleepbeweging wilt afhandelen, moet u eerst het standaardgedrag annuleren. Luister naar de gebeurtenissen `dragenter` en `dragover` van alle HTML-elementen die u wilt gebruiken als neerzetdoelen. Roep in de handlers voor deze gebeurtenissen de methode `preventDefault()` van het verzonden gebeurtenisobject aan. Als u het standaardgedrag annuleert, mogen niet-bewerkbare gebieden neergezette gegevens ontvangen.

Neergezette gegevens ophalen

Adobe AIR 1.0 of hoger

U kunt de neergezette gegevens ophalen in de handler voor de gebeurtenis `ondrop`:

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

Gebruik de methode `dataTransfer.getData()` om de gegevens op het klembord te lezen en geef daarbij het MIME-type door van de gegevensindeling die u wilt lezen. Met de eigenschap `types` van het object `dataTransfer` kunt u te weten komen welke gegevensindelingen beschikbaar zijn. De array `types` bevat de MIME-typetekenreeks van elke beschikbare indeling.

Wanneer u het standaardgedrag in de gebeurtenis dragenter of dragover annuleert, bent u er verantwoordelijk voor dat neergezette gegevens op de juiste plaats in het document worden ingevoegd. Er bestaat geen API om een muispositie te converteren naar een invoegpositie in een element. Hierdoor kan het moeilijk zijn om sleepbewegingen voor invoegbewerkingen te implementeren.

Voorbeeld: het standaardgedrag voor het inslepen van HTML overschrijven

Adobe AIR 1.0 of hoger

In dit voorbeeld wordt een neerzetdoel geïmplementeerd waarmee een tabel wordt weergegeven met elke gegevensindeling die beschikbaar is in het neergezette item.

Het standaardgedrag wordt gebruikt, waardoor tekst, koppelingen en afbeeldingen in de toepassing mogen worden gesleept. In dit voorbeeld wordt het standaardgedrag voor inslepen overschreven voor het div-element dat optreedt als neerzetdoel. Als u wilt dat niet-bewerkbare inhoud een insleepbeweging accepteert, is het belangrijk dat u de methode `preventDefault()` oproept van het gebeurtenisobject dat wordt verzonden voor de gebeurtenissen `dragenter` en `dragover`. Als reactie op de gebeurtenis `drop` converteert de handler de overgedragen gegevens naar een HTML-rijelement en wordt deze rij ingevoegd in een tabel die wordt weergegeven.

```
 <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>
```

Bestanden neerzetten in niet-toepassingssandboxen in HMTL

Adobe AIR 1.0 of hoger

Inhoud die niet van de toepassing afkomstig is, heeft geen toegang tot File-objecten die worden gemaakt wanneer bestanden naar een AIR-toepassing worden gesleept. Het is ook niet mogelijk een van deze File-objecten aan inhoud van de toepassing door te geven via een sandboxbridge. (De objecteigenschappen moeten worden benaderd tijdens het serialiseren.) U kunt echter wel bestanden in uw toepassing neerzetten door te luisteren naar de nativeDragDrop-gebeurtenissen van AIR voor het HTMLLoader-object.

Als een gebruiker een bestand neerzet in een frame dat inhoud bevat die niet van de toepassing afkomstig is, wordt de neerzetgebeurtenis gewoonlijk niet van het onderliggende naar het bovenliggende item doorgevoerd. Omdat de gebeurtenissen die door de HTMLLoader (die de container is voor alle HTML-inhoud in een AIR-toepassing) worden verzonden geen deel uitmaken van de HTML-gebeurtenissenstroom, kunt u de neerzetgebeurtenis echter nog steeds ontvangen in inhoud van de toepassing.

Om de gebeurtenis voor het neerzetten van een bestand te ontvangen, voegt het bovenliggende document een gebeurtenislistener aan het HTMLLoader-object toe met de verwijzing die is geleverd door `window.htmlLoader`:

```
window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

In het volgende voorbeeld laadt een bovenliggend document een onderliggende pagina in een externe sandbox (http://localhost/). Het bovenliggende document luistert naar de gebeurtenis `nativeDragDrop` van het HTMLLoader-object en haalt hier de bestands-URL uit op.

```
<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>
```

Het onderliggende document moet een geldig doel opleveren door de `preventDefault()`-methode van het gebeurtenisobject aan te roepen in de HTML-gebeurtenishandlers `dragenter` en `dragover`. Anders kan het neerzetten niet plaatsvinden.

```
<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>
```

Bestandsbeloften neerzetten

Adobe AIR 2 of hoger

Een bestandspromise is een klembordindeling voor verslepen en neerzetten waarmee een gebruiker een bestand kan verslepen die nog niet buiten een AIR-toepassing bestaat. Met een bestandsbelofte kunt u bijvoorbeeld toestaan dat een gebruiker een proxypictogram naar een bureaubladmap sleept. Het proxypictogram representeert een bestand of gegevens die via een URL beschikbaar zijn. Nadat de gebruiker het pictogram heeft neergezet, worden de gegevens gedownload en wordt het bestand geschreven naar de locatie waar de bestandsbelofte is neergezet.

Gebruik de URLFilePromise-klasse in een AIR-toepassing om bestanden die via een URL toegankelijk zijn, te slepen en neer te zetten. De URLFilePromise-implementatie is opgenomen in de aircore-bibliotheek als onderdeel van de SDK van AIR 2. Gebruik ofwel het bestand aircore.swc of aircore.swf in de map SDK frameworks/libs/air.

U kunt ook uw eigen FilePromise-logica implementeren met de IFilePromise-interface (gedefinieerd in het runtime flash.desktop-pakket).

Het concept van bestandsbeloften is vergelijkbaar met uitgestelde rendering aan de hand van een gegevenshandlerfunctie op het klembord. Gebruik bij het slepen en neerzetten van bestanden bestandsbeloften in plaats van uitgestelde rendering. De techniek voor uitgestelde rendering kan bij het genereren of downloaden van gegevens leiden tot ongewenste pauzes in de sleepbeweging. Gebruik uitgestelde rendering voor kopiëren en plakken (hiervoor is geen ondersteuning bij bestandsbeloften).

Beperkingen voor bestandsbeloften

Bestandsbeloften hebben de volgende beperkingen vergeleken met andere gegevensindelingen die u kunt plaatsen op het klembord voor slepen en neerzetten:

- Bestandsbeloften kunnen alleen uit een AIR-toepassing worden gesleept. Ze kunnen niet in een AIR-toepassing worden neergezet.
- Bestandsbeloften worden niet op alle besturingssystemen ondersteund. Gebruik de eigenschap `Clipboard.supportsFilePromise` om te testen of bestandsbeloften op het hostsysteem worden ondersteund. Op systemen die geen ondersteuning bieden voor bestandsbeloften, moet u een alternatief mechanisme bieden voor het downloaden of het genereren van de bestandsgegevens.
- Bestandsbeloften kunnen niet worden gebruikt op het klembord voor kopiëren en plakken (`Clipboard.generalClipboard`).

Meer Help-onderwerpen

flash.desktop.IFilePromise

air.desktop.URLFilePromise

Externe bestanden neerzetten

Adobe AIR 2 of hoger

Gebruik de URLFilePromise-klasse om bestandsbelofteobjecten te maken die via een URL beschikbare bestanden of gegevens representeren. Met de klembordindeling `FILE_PROMISE_LIST` voegt u een of meer bestandsbelofteobjecten toe aan het klembord. In het volgende voorbeeld wordt een bestand dat beschikbaar is via http://www.example.com/foo.txt, gedownload en opgeslagen als bar.txt op de locatie waar de bestandsbelofte is neergezet. (De namen van het externe bestand en het lokale bestand hoeven niet hetzelfde te zijn.)

```
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 );
}
```

U kunt de gebruiker toestaan om meer dan een bestand tegelijkertijd te slepen door bestandsbelofteobjecten toe te voegen aan de array die aan het klembord is toegewezen. U kunt ook submappen opgeven in de eigenschap `relativePath` zodat sommige of alle bestanden die deel uitmaken van de bewerking, in een submap worden geplaatst die relatief is aan de locatie waar de bestandsbelofte is neergezet.

Het volgende voorbeeld illustreert hoe u een sleepbewerking met meerdere bestandsbeloften kunt uitvoeren. In dit voorbeeld wordt de HTML-pagina *article.html* als bestandsbelofte op het klembord geplaatst, evenals twee bijbehorende gekoppelde afbeeldingsbestanden. De afbeeldingen worden gekopieerd naar een submap *images*, zodat de relatieve koppelingen behouden blijven.

```
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 );
}
```

Implementatie van de IFilePromise-interface

Adobe AIR 2 of hoger

Als u bestandsbeloften beschikbaar wilt stellen voor resources die niet via een URLFilePromise-object toegankelijk zijn, kunt u de IFilePromise-interface in een aangepaste klasse implementeren. De IFilePromise-interface definieert de methoden en eigenschappen die in AIR worden gebruikt om toegang te krijgen tot de gegevens die naar een bestand moeten worden geschreven als de bestandsbelofte eenmaal is neergezet.

Een IFilePromise-implementatie geeft in AIR nog een object door dat de gegevens voor de bestandsbelofte levert. De IDataInput-interface moet door dit object worden geïmplementeerd. Deze interface wordt in AIR gebruikt om de gegevens te lezen. De URLFilePromise-klasse bijvoorbeeld, waardoor IFilePromise wordt geïmplementeerd, gebruikt een URLStream-object als de gegevensaanbieder.

AIR kan de gegevens synchroon of asynchroon lezen. De IFilePromise-implementatie rapporteert welke toegangsmodus wordt ondersteund door de juiste waarde in de eigenschap `isAsync` te retourneren. In geval van asynchrone gegevenstoegang moet het gegevensaanbiederobject de IEventDispatcher-interface implementeren en de benodigde gebeurtenissen verzenden, zoals `open`, `progress` en `complete`.

U kunt een aangepaste klasse of een van de volgende ingebouwde klassen gebruiken als gegevensaanbieder voor een bestandsbelofte:

- ByteArray (synchroon)
- FileStream (synchroon of asynchroon)
- Socket (asynchroon)
- URLStream (asynchroon)

Om de IFilePromise-interface te implementeren moet u code voor de volgende functies en eigenschappen leveren:

- `open():IDataInput` - Retourneert het gegevensaanbiederobject waaruit de gegevens voor de bestandsbelofte worden gelezen. De IDataInput-interface moet door dit object worden geïmplementeerd. In geval van asynchrone gegevens moet dit object eveneens de IEventDispatcher-interface implementeren en de benodigde gebeurtenissen verzenden (zie "Een asynchrone gegevensaanbieder gebruiken in een bestandsbelofte" op pagina 663).
- `get relativePath():String` - Levert het pad, waaronder de bestandsnaam, voor het gemaakte bestand. Het pad wordt omgezet, relatief aan de locatie die door de gebruiker tijdens het slepen en neerzetten is gekozen. Gebruik de constante `File.separator` als u paden met mappen opgeeft. Op deze manier weet u zeker dat in het pad het juiste scheidingsteken voor het hostbesturingssysteem wordt gebruikt. Als u wilt toestaan dat het pad tijdens runtime wordt ingesteld, voegt u een functie van het type setter toe of gebruikt u een constructorparameter.
- `get isAsync():Boolean` - Informeert AIR of het gegevensaanbiederobject gegevens asynchroon of synchroon levert.
- `close():void` - Wordt aangeroepen als de gegevens volledig zijn gelezen (of als een fout voorkomt dat er verder wordt gelezen). U kunt deze functie gebruiken voor het opschonen van resources.
- `reportError( e:ErrorEvent ):void` - Wordt aangeroepen als er zich bij het lezen van de gegevens een fout voordoet.

Alle IFilePromise-methoden worden aangeroepen bij het slepen en neerzetten van een bestandsbelofte. Doorgaans mogen deze methoden niet rechtstreeks door uw toepassingslogica worden aangeroepen.

Een synchrone gegevensaanbieder gebruiken in een bestandsbelofte

Adobe AIR 2 of hoger

U kunt de IFilePromise-interface het eenvoudigst implementeren met een synchroon gegevensaanbiederobject, zoals een ByteArray of een synchrone FileStream. In het volgende voorbeeld wordt een ByteArray-object gemaakt, gevuld met gegevens en geretourneerd wanneer de methode `open()` wordt aangeroepen.

```
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 );
        }
    }
}
```

In de praktijk is het nut van synchrone bestandsbeloften beperkt. Als de hoeveelheid gegevens klein is, kunt u net zo gemakkelijk een bestand in een tijdelijke map maken en een normale bestandenlijstarray aan het klembord voor slepen en neerzetten toevoegen. Aan de andere kant, als de hoeveelheid gegevens groot is of als de computer zwaar wordt belast door het genereren van de gegevens, is een lang synchroon proces nodig. Lange synchrone processen kunnen updates van de gebruikersinterface gedurende een aanzienlijke periode blokkeren. Hierdoor kan het lijken alsof uw toepassing niet reageert. Om dit probleem te vermijden, kunt u een asynchrone gegevensaanbieder maken die wordt aangestuurd door een timer.

Een asynchrone gegevensaanbieder gebruiken in een bestandsbelofte

Adobe AIR 2 of hoger

Als u een asynchroon gegevensaanbiederobject gebruikt, moet de eigenschap `isAsync` van IFilePromise zijn ingesteld op `true` en moet het object dat wordt geretourneerd door de methode `open()` de IEventDispatcher-interface implementeren. Tijdens runtime wordt er geluisterd of een aantal alternatieve gebeurtenissen plaatsvinden, zodat verschillende ingebouwde objecten als gegevensaanbieder kunnen worden gebruikt. `progress`-gebeurtenissen worden bijvoorbeeld verzonden door FileStream- en URLStream-objecten, terwijl `socketData`-gebeurtenissen worden verzonden door Socket-objecten. Tijdens runtime wordt er geluisterd of al deze objecten de juiste gebeurtenissen verzenden.

Het lezen van gegevens in het gegevensaanbiederobject wordt door de volgende gebeurtenissen aangestuurd:

- Event.OPEN - Meldt dat de gegevensbron klaar is.
- ProgressEvent.PROGRESS - Meldt dat de gegevens beschikbaar zijn. Tijdens runtime wordt de hoeveelheid beschikbare gegevens in het gegevensaanbiederobject gelezen.
- ProgressEvent.SOCKET_DATA - Meldt dat de gegevens beschikbaar zijn. De gebeurtenis `socketData` wordt verzonden door op sockets gebaseerde objecten. Voor andere typen objecten moet u een `progress`-gebeurtenis verzenden. (Tijdens runtime wordt er geluisterd wanneer beide gebeurtenissen plaatsvinden, zodat er kan worden bepaald wanneer er gegevens kunnen worden gelezen.)
- Event.COMPLETE - Meldt dat de gegevens helemaal gelezen zijn.
- Event.CLOSE - Meldt dat de gegevens helemaal gelezen zijn. (Tijdens runtime wordt er geluisterd wanneer `close` en `complete` hiervoor plaatsvinden.)
- IOErrorEvent.IOERROR - Meldt dat er zich bij het lezen van de gegevens een fout heeft voorgedaan. Het maken van een bestand wordt afgebroken en de methode `close()` van IFilePromise wordt aangeroepen.
- SecurityErrorEvent.SECURITY_ERROR - Meldt dat er zich een beveiligingsfout heeft voorgedaan. Het maken van een bestand wordt afgebroken en de methode `close()` van IFilePromise wordt aangeroepen.
- HTTPStatusEvent.HTTP_STATUS - Wordt samen met `httpResponseStatus` gebruikt om ervoor te zorgen dat de beschikbare gegevens de gewenste inhoud representeren en niet een foutbericht (zoals een 404-pagina). Objecten die op het HTTP-protocol zijn gebaseerd, moeten deze gebeurtenis verzenden.
- HTTPStatusEvent.HTTP_RESPONSE_STATUS - Wordt samen met `httpStatus` gebruikt om ervoor te zorgen dat de beschikbare gegevens de gewenste inhoud representeren. Objecten die op het HTTP-protocol zijn gebaseerd, moeten deze gebeurtenis verzenden.

De gegevensaanbieder moet deze gebeurtenissen in de volgende volgorde verzenden:

1. De gebeurtenis `open`
2. De gebeurtenis `progress` of de gebeurtenis `socketData`
3. De gebeurtenis `complete` of de gebeurtenis `close`

Opmerking: *De ingebouwde objecten FileStream, Socket en URLStream verzenden automatisch de juiste gebeurtenissen.*

In het volgende voorbeeld wordt een bestandsbelofte gemaakt met een aangepaste asynchrone gegevensaanbieder. De gegevensaanbiederklasse biedt ByteArray uit (voor de ondersteuning voor IDataInput) en implementeert de IEventDispatcher-interface. Bij elke timer-gebeurtenis genereert het object een hoeveelheid gegevens en verzendt het object een progress-gebeurtenis om tijdens runtime te melden dat de gegevens beschikbaar zijn. Als er genoeg gegevens zijn gegenereerd, verzendt het object een complete-gebeurtenis.

```
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 );
    }
}
}
```

Opmerking: *Omdat de AsyncDataProvider-klasse in het voorbeeld ByteArray uitbreidt, kan deze klasse niet tevens EventDispatcher uitbreiden. Om de IEventDispatcher-interface te implementeren, gebruikt de klasse een intern EventDispatcher-object en stuurt de IEventDispatcher-methode aanroepen door naar dat interne object. U kunt eventueel ook EventDispatcher uitbreiden en IDataInput implementeren (of beide interfaces implementeren).*

De asynchrone IFilePromise-implementatie is bijna identiek aan de synchrone implementatie. De belangrijkste verschillen zijn dat `isAsynctrue` retourneert en dat de methode `open()` een asynchroon gegevensobject retourneert:

```
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 );
        }
    }
}
```

Hoofdstuk 36: Werken met menu's

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruik de klassen in de contextmenu-API om de contextmenu's in de webtoepassingen Flex en Flash Player aan te passen.

Gebruik de klassen in de native menu-API om de toepassing, het venster, de context en pop-upmenu's te definiëren in Adobe® AIR®.

Basisbeginselen van menu's

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Lees voor een snelle uitleg en codevoorbeelden van het maken van native menu's in AIR-toepassingen, de volgende snelstartartikelen in de Adobe Developer Connection:

- Native menu's toevoegen aan een AIR-toepassing (Flex)
- Native menu's toevoegen aan een AIR-toepassing (Flash)

Met de native menuklassen hebt u toegang tot de native menufuncties van het besturingssysteem waarin uw toepassing wordt uitgevoerd. U kunt NativeMenu-objecten gebruiken voor toepassingsmenu's (beschikbaar in Mac OS X), venstermenu's (beschikbaar in Windows en Linux), snelmenu's en pop-upmenu's.

Buiten AIR, kunt u contextmenuklassen gebruiken gebruiken om het contextmenu aan te passen dat automatisch door Flash Player wordt weergegeven wanneer een gebruiker met de rechtermuisknop of met cmd op een object in uw toepassing klikt. (Er wordt geen automatisch contextmenu weergegeven voor AIR-toepassingen.)

Menu-klassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De menuklassen zijn:

Pakket	Klassen
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

Menutypen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

AIR ondersteunt de volgende menutypen:

Contextmenu's Contextmenu's worden geopend als reactie op klikken met de rechtermuisknop of klikken met de Command-toets ingedrukt op een interactief object in SWF-inhoud of een documentelement in HTML-inhoud.

In de Flash Player runtime wordt automatisch een contextmenu weergegeven. U kunt met de klassen ContextMenu en ContextMenuItem uw opdrachten aan het menu toevoegen. U ook een aantal, maar niet alle, ingebouwde opdrachten verwijderen.

U kunt in AIR-runtime een contextmenu maken met behulp van de klasse NativeMenu of ContextMenu. In HTML-inhoud in AIR kunt u met de API's voor Webkit HTML en JavaScript contextmenu's toevoegen aan HTML-elementen.

Toepassingsmenu's (alleen AIR) Een toepassingsmenu is een globaal menu dat geldt voor de gehele toepassing. Toepassingsmenu's worden ondersteund in Mac OS X, maar niet in Windows of Linux. In Mac OS X maakt het besturingssysteem automatisch een toepassingsmenu. U kunt de AIR API voor menu's gebruiken om items en submenu's toe te voegen aan de standaardmenu's. U kunt listeners toevoegen om de bestaande menuopdrachten af te handelen. U kunt ook bestaande items verwijderen.

Venstermenu's (alleen AIR) Een venstermenu is gekoppeld aan één venster en wordt weergegeven onder de titelbalk. U kunt menu's aan een venster toevoegen door een NativeMenu-object te maken en dit toe te wijzen aan de eigenschap `menu` van het NativeWindow-object. Venstermenu's worden ondersteund in de besturingssystemen Windows en Linux, maar niet in Mac OS X. Native venstermenu's kunnen alleen worden gebruikt met vensters die systeemchroom hebben.

Pictogrammenu's voor koppelings- en systeemtrays (alleen AIR) Deze pictogrammenu's zijn vergelijkbaar met snelmenu's en worden toegewezen aan een toepassingspictogram in de dok van Mac OS X of het Windows- en Linux-systeemvak op de taakbalk. Dock- en systeemvakpictogrammenu's maken gebruik van de klasse NativeMenu. In Mac OS X worden de items in het menu toegevoegd boven de standaarditems van het besturingssysteem. In Windows of Linux is er geen standaardmenu.

Pop-upmenu's (alleen AIR) Een pop-upmenu van AIR is vergelijkbaar met een contextmenu, maar is niet noodzakelijkerwijs gekoppeld aan een bepaald toepassingsobject of -onderdeel. U kunt pop-upmenu's op een willekeurige plaats in een venster weergeven door de methode `display()` van een willekeurig NativeMenu-object op te roepen.

Aangepaste menu's Native menu's worden volledig door het besturingssysteem samengesteld en bestaan dus buiten de Flash- en HTML-renderingmodellen. In plaats van native menu's kunt u altijd uw eigen aangepaste, non-native menu's maken met MXML, ActionScript of JavaScript (in AIR). Zulke menu's moeten volledig binnen toepassingsinhoud worden gerenderd.

Flex-menu's Het Adobe® Flex™-framework biedt een set Flex-menucomponenten. De Flex-menu's worden door de runtime getekend in plaats van het besturingssysteem en zijn geen *native* menu's. Een Flex-menucomponent kan worden gebruikt voor Flex-vensters die niet de systeeminterface gebruiken. Een ander voordeel van het gebruik van Flex-menucomponenten is dat u menu's declaratief kunt opgeven in MXML-indeling. Als u het Flex-framework gebruikt, gebruikt u de Flex-menuklassen voor venstermenu's in plaats van de native klassen.

Standaardmenu's (alleen AIR)

De volgende standaardmenu's worden geleverd door het besturingssysteem of door een ingebouwde AIR-klasse:

- Toepassingsmenu in Mac OS X
- Dock-pictogrammenu in Mac OS X

- Contextmenu voor geselecteerde tekst en afbeeldingen in HTML-inhoud
- Contextmenu voor geselecteerde tekst in een TextField-object (of een object dat een uitbreiding vormt voor TextField)

Informatie over contextmenu's

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In SWF-inhoud kan een object dat gegevens overneemt van InteractiveObject, een contextmenu krijgen. Hiervoor wijst u een menuobject toe aan de eigenschap `contextMenu` ervan. Een aantal opdrachten is standaard aanwezig, zoals Volgende, Vorige, Afdrukken, Kwaliteit en In-/uitzoomen. In de AIR runtime kan het menuobject dat is toegewezen aan `contextMenu` van het type NativeMenu of ContextMenu zijn. In de Flash Player runtime is alleen de Contextmenu-variabele beschikbaar.

Wanneer u de klassen ContextMenu en ContextMenuItem gebruikt, kunt u luisteren naar native menugebeurtenissen of contextmenugebeurtenissen; beide worden verzonden. Een van de voordelen van de eigenschappen van het ContextMenuEvent-object is dat contextMenuOwner het object identificeert waaraan het menu is gekoppeld, en dat `mouseTarget` het object identificeert waarop werd geklikt om het menu te openen. Deze informatie is niet beschikbaar bij het NativeMenuEvent-object.

In het volgende voorbeeld wordt een Sprite-object gemaakt en wordt er een eenvoudig contextmenu voor bewerken aan toegevoegd:

```
 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");}
```

Opmerking: *In tegenstelling tot SWF-inhoud die wordt weergegeven in een browseromgeving, bezitten contextmenu's in AIR geen ingebouwde opdrachten.*

Een Flash Player-contextmenu aanpassen

In een browser of projector hebben contextmenu's in SWF-inhoud altijd ingebouwde items. U kunt al deze standaardopdrachten uit het menu verwijderen, behalve de opdrachten Instellingen en Over. Als de werkgebiedeigenschap `showDefaultContextMenu` wordt ingesteld op `false`, worden deze opdrachten uit het contextmenu verwijderd.

Als u een aangepast contextmenu voor een specifiek weergaveobject wilt maken, maakt u een nieuwe instantie van de klasse ContextMenu, roept u de methode `hideBuiltInItems()` aan en wijst u die instantie toe aan de eigenschap `contextMenu` van die instantie DisplayObject. In het volgende voorbeeld wordt een dynamisch getekend vierkant gemaakt, met een opdracht in het contextmenu om de kleur ervan in een willekeurige kleur te wijzigen:

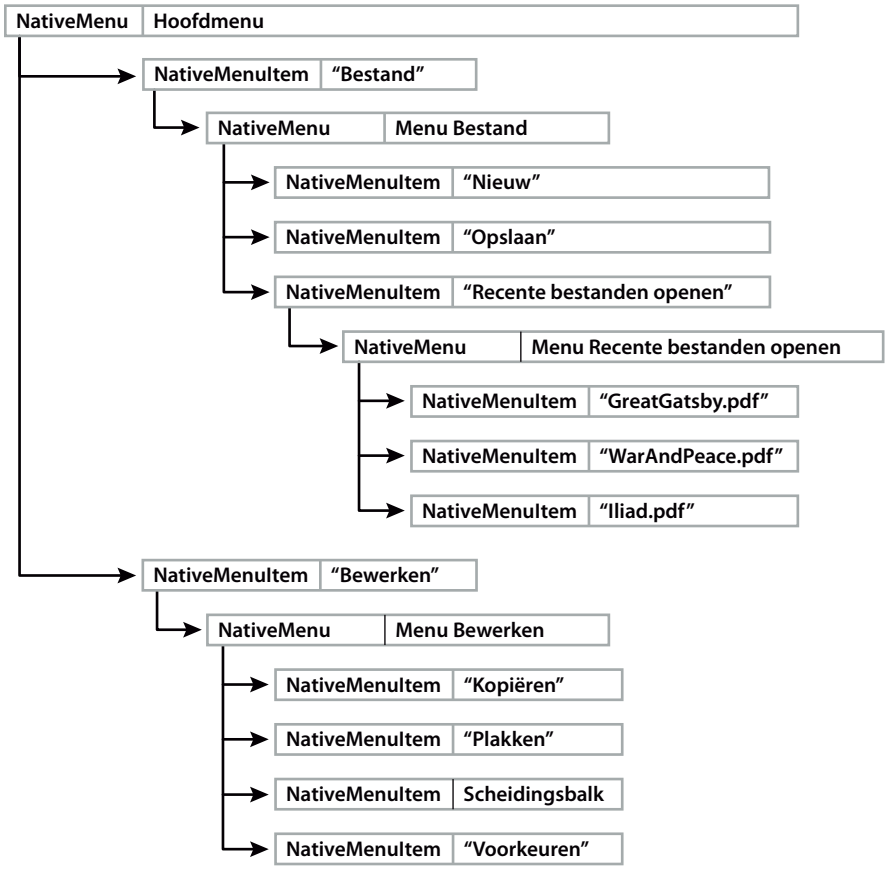
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

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);
}
```

Native menustructuur (AIR)

Adobe AIR 1.0 of hoger

Native menu's zijn hiërarchisch. NativeMenu-objecten bevatten onderliggende NativeMenuItem-objecten. NativeMenuItem-objecten die submenu's vertegenwoordigen, kunnen op hun beurt weer NativeMenu-objecten bevatten. Het menuobject van het hoogste niveau (hoofdniveau) in de structuur vertegenwoordigt de menubalk voor toepassings- en venstermenu's. (Context-, pictogram- en pop-upmenu's hebben geen menubalk.)

In het volgende diagram wordt de structuur van een typisch menu geïllustreerd. Het hoofdmenu vertegenwoordigt de menubalk en bevat twee menu-items die verwijzen naar de submenu's *File* en *Edit*. Het submenu File in deze structuur bevat twee opdrachtitems en een item dat verwijst naar het submenu *Open Recent Menu*, dat zelf weer drie items bevat. Het submenu Edit bevat drie opdrachten en een scheidingslijn.



Voor de definitie van een submenu is zowel een NativeMenu- als een NativeMenuItem-object nodig. Het NativeMenuItem-object definieert het label dat wordt weergegeven in het bovenliggende menu en stelt de gebruiker in staat het submenu te openen. Het NativeMenu-object fungeert als container voor items in het submenu. Het NativeMenuItem-object verwijst naar het NativeMenu-object via de eigenschap `submenu` van het NativeMenuItem.

Zie "Voorbeeld van native menu: venster- en toepassingsmenu (AIR)" op pagina 679 voor een codevoorbeeld waarin dit menu wordt gemaakt).

Menugebeurtenissen

Adobe AIR 1.0 of hoger

NativeMenu- en NativeMenuItem-objecten verzenden allebei `preparing`-, `displaying`- en `select`-gebeurtenissen:

Preparing: wanneer het object op het punt staat een gebruikersinteractie in gang te zetten, verzenden het menu en de bijbehorende menu-items een `preparing`-gebeurtenis naar alle geregistreerde listeners. Tot interactie wordt het openen van het menu of het selecteren van een item met een sneltoets gerekend.

Opmerking: *De `preparing`-gebeurtenis is alleen beschikbaar voor Adobe AIR 2.6 en latere versies.*

Displaying: vlak voordat een menu wordt weergegeven, verzenden het menu en de bijbehorende menu-items de gebeurtenis `displaying` naar alle geregistreerde listeners.

De gebeurtenissen `preparing` en `displaying` geven u de mogelijkheid om de menu-inhoud of de weergave van items bij te werken voordat de gebruiker deze ziet. U kunt bijvoorbeeld in de listener voor de gebeurtenis `displaying` van het menu "Open Recent" de menu-items wijzigen zodat de lijst van onlangs bekeken documenten wordt weergegeven.

Als u het menu-item verwijdert waarvan de sneltoets een `preparing`-gebeurtenis heeft geactiveerd, wordt de menu-interactie in feite geannuleerd en wordt geen `select`-gebeurtenis verzonden.

De `target`- en `currentTarget`-eigenschappen van de gebeurtenis zijn beide het object waarop de listener is geregistreerd: het menu zelf of een van de menu-items.

De `preparing`-gebeurtenis wordt verzonden vóór de `displaying`-gebeurtenis. Over het algemeen wordt naar een van de twee gebeurtenissen geluisterd, niet naar allebei.

Select: Wanneer de gebruiker een opdrachtitem kiest, verzendt dat item de gebeurtenis `select` naar eventuele geregistreerde listeners. Submenu- en scheidingslijnitems kunnen niet worden geselecteerd en verzenden dus nooit de gebeurtenis `select`.

De gebeurtenis `select` stroomt omhoog van een menu-item naar het menu dat dit item bevat, of helemaal naar het hoofdmenu. U kunt luisteren naar `select`-gebeurtenissen rechtstreeks op een item, of hogerop in de menustructuur. Wanneer u luistert naar de gebeurtenis `select` op een menu, kunt u het geselecteerde item identificeren met behulp van de eigenschap `target` van de gebeurtenis. Terwijl de gebeurtenis omhoogstroomt door de menuhiërarchie, identificeert de eigenschap `currentTarget` van het gebeurtenisobject het huidige menuobject.

Opmerking: *ContextMenu- en ContextMenuItem-objecten verzenden `menuItemSelect`- en `menuSelect`-gebeurtenissen, evenals `select`-, `preparing`- en `displaying`-gebeurtenissen.*

Hoofdalternatieven voor native menu-opdrachten (AIR)

Adobe AIR 1.0 of hoger

U kunt een toetsequivalent (sneltoets) toewijzen aan een menuopdracht. Het menu-item verzendt de gebeurtenis `select` naar alle geregistreerde listeners wanneer de toets of toetsencombinatie wordt ingedrukt. De opdracht wordt alleen opgeroepen wanneer het menu dat het item bevat, deel uitmaakt van het menu van de toepassing of het actieve venster.

Toetsequivalenten bestaan uit twee delen: een tekenreeks die de primaire toets vertegenwoordigt en een array van wijzigingstoetsen die ook moeten worden ingedrukt. Om de primaire toets toe te wijzen, stelt u de eigenschap `keyEquivalent` voor het menu-item in op de uit één teken bestaande tekenreeks voor die toets. Als u een hoofdletter gebruikt, wordt de Shift-toets automatisch toegevoegd aan de wijzigingsarray.

In Mac OS X is de standaard wijzigingstoets de Command-toets (`Keyboard.COMMAND`). In Windows en Linux is het de Ctrl-toets (`Keyboard.CONTROL`). Deze standaardtoetsen worden automatisch toegevoegd aan de wijzigingsarray. Als u andere wijzigingstoetsen wilt toewijzen, wijst u een nieuwe array met de gewenste toetscodes toe aan de eigenschap `keyEquivalentModifiers`. De standaardarray wordt dan overschreven. Of u nu de standaard wijzigingstoetsen gebruikt dan wel een eigen wijzigingsarray toewijst, de Shift-toets wordt altijd toegevoegd wanneer de tekenreeks die u toewijst aan de eigenschap `keyEquivalent` een hoofdletter is. Constanten voor de toetscodes die moeten worden gebruikt voor de wijzigingstoetsen, worden gedefinieerd in de klasse Keyboard.

De toegewezen tekenreeks voor het toetsequivalent wordt automatisch weergegeven naast de naam van het menu-item. De indeling is afhankelijk van het besturingssysteem van de gebruiker en de systeemvoorkeuren.

Opmerking: *Als u in het Windows-besturingssysteem de waarde* `Keyboard.COMMAND` *toewijst aan een toetswijzigingsarray, wordt er geen toetsequivalent weergegeven in het menu. De Control-toets moet dan worden gebruikt om de menuopdracht te activeren.*

In het volgende voorbeeld wordt `Ctrl+Shift+G` toegewezen als toetsequivalent voor een menu-item:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

In dit voorbeeld wijst u `Ctrl+Shift+G` als toetsequivalent toe door de wijzigingsarray rechtstreeks in te stellen:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

Opmerking: *Toetsequivalenten worden alleen getriggerd voor toepassings- en venstermenu's. Als u een toetsequivalent toevoegt aan een context- of pop-upmenu, wordt het toetsequivalent weergegeven in het menulabel. De daaraan gekoppelde menuopdracht wordt echter nooit opgeroepen.*

Mnemonics (AIR)

Adobe AIR 1.0 of hoger

Mnemonics (toetsenbordsneltoetsen in de vorm van onderstreepte letters) maken deel uit van de besturingssysteem-toetsenbordinterface voor menu's. In Linux, Mac OS X en Windows kunnen gebruikers menu's openen en opdrachten selecteren met het toetsenbord; er zijn echter kleine verschillen.

In Mac OS X typen de gebruikers de eerste letter of eerste twee letters van het menu of de opdracht waarna op Enter wordt gedrukt. De eigenschap `mnemonicIndex` wordt genegeerd.

In Windows is maar één letter significant. Standaard is de significante letter het eerste teken van het label. Als u echter een mnemonic toewijst aan het menu-item, wordt de desbetreffende letter het significante teken. Als twee items in een menu hetzelfde significante teken hebben (ongeacht of een mnemonic is toegewezen), verandert de interactie tussen het toetsenbord en het menu enigszins. De gebruiker kan nu niet één letter indrukken om het menu of de opdracht te selecteren, maar moet de letter zo vaak als nodig indrukken om het gewenste item te markeren. Vervolgens moet de gebruiker op Enter drukken om de selectie te voltooien. U behoudt een consistente werking door een unieke mnemonic aan elk item in een menu voor venstermenu's toe te wijzen.

In Linux is geen standaardmnemonic opgegeven. Als u een mnemonic wilt opgeven, moet u een waarde opgeven voor de eigenschap `mnemonicIndex` van een menu-item.

Geef de gewenste mnemonic op als index bij de labeltekenreeks. De index van het eerste teken van een label is 0. Als u bijvoorbeeld “r” wilt gebruiken als mnemonic voor het menu-item met het label “Format”, stelt u de eigenschap `mnemonicIndex` in op 2.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

Status van het menu-item

Adobe AIR 1.0 of hoger

Menu-items hebben de volgende twee statuseigenschappen: `checked` en `enabled`:

checked Stel deze in op `true` als u een vinkje wilt weergeven naast het itemlabel.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

enabled Schakel deze waarde tussen `true` en `false` om aan te geven of de opdracht is ingeschakeld. Uitgeschakelde items worden grijs weergegeven en verzenden geen gebeurtenis `select`.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

Objecten koppelen aan een menu-item

Adobe AIR 1.0 of hoger

Met de eigenschap `data` van de klasse NativeMenuItem kunt u in ieder item verwijzen naar een willekeurig object. Zo kunt u in een menu met recent geopende bestanden bijvoorbeeld het File-object voor ieder document toewijzen aan ieder menu-item.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

Native menu's maken (AIR)

Adobe AIR 1.0 of hoger

In dit onderwerp wordt beschreven hoe u de verschillende soorten native menu's kunt maken die door AIR worden ondersteund.

Hoofdmenuobjecten maken

Adobe AIR 1.0 of hoger

Als u een NativeMenu-object wilt maken dat fungeert als basis van het menu, gebruikt u de NativeMenu-constructor:

```
var root:NativeMenu = new NativeMenu();
```

Voor toepassings- en venstermenu's vertegenwoordigt het hoofdmenu de menubalk. Dit menu moet alleen items bevatten die submenu's openen. Contextmenu's en pop-upmenu's bezitten geen menubalk. Het hoofdmenu kan dus zowel opdrachten en scheidingslijnen bevatten als submenu's.

Nadat u het menu hebt gemaakt, kunt u menu-items toevoegen. Items worden in het menu weergegeven in de volgorde waarin ze zijn toegevoegd, behalve als u de items bij een specifieke index toevoegt met behulp van de methode `addItemAt()` van een menuobject.

U kunt het menu toewijzen als toepassings-, venster-, pictogram- of contextmenu. U kunt het ook weergeven als pop-upmenu. Dit wordt beschreven in de volgende secties:

Het toepassingsmenu instellen of het venstermenu instellen
Het is belangrijk dat uw code kan worden gebruikt voor zowel toepassingsmenu's (ondersteund door Mac OS) als venstermenu's (ondersteund op andere besturingssystemen).

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

Opmerking: *Mac OS definieert een menu met standaarditems voor iedere toepassing. Als u een nieuw NativeMenu-object toewijst aan de eigenschap* `menu` *van het NativeApplication-object, wordt het standaardmenu vervangen. U kunt er ook voor kiezen het standaardmenu te gebruiken in plaats van het te vervangen.*

Adobe Flex biedt een FlexNativemenu-klasse zodat u gemakkelijk menu's kunt maken die op verschillende besturingssystemen werken. Gebruikt u het Flex-framework, gebruik dan de FlexNativeMenu-klassen in plaats van de NativeMenu-klasse.

Contextmenu's instellen op een interactief object

```
interactiveObject.contextMenu = root;
```

Dock-pictogrammenu's instellen of systeemvakpictogrammen instellen

Het is belangrijk dat uw code kan worden gebruikt voor zowel toepassingsmenu's (ondersteund door Mac OS) als venstermenu's (ondersteund op andere besturingssystemen).

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

Opmerking: *Mac OS X definieert een standaardmenu voor het dock-pictogram van de toepassing. Wanneer u een nieuw NativeMenu toewijst aan de menu-eigenschap van het DockIcon-object, worden de items in dat menu weergegeven boven de standaarditems. U kunt de standaard menu-items niet verwijderen, openen of wijzigen.*

Menu's weergeven als pop-up

```
root.display(stage, x, y);
```

Meer Help-onderwerpen

Developing cross-platform AIR applications (AIR-toepassingen voor meerdere platformen ontwikkelen)

Submenu's maken

Adobe AIR 1.0 of hoger

Als u een submenu wilt maken, voegt u een NativeMenuItem-object toe aan het bovenliggende menu. Vervolgens wijst u het NativeMenu-object dat het submenu definieert, toe aan de eigenschap `submenu` van het item. AIR biedt u twee manieren om submenu-items en de bijbehorende menuobjecten te maken:

U kunt een menu-item en het bijbehorende menuobject in één stap maken met behulp van de methode `addSubmenu()`:

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

U kunt ook het menu-item maken en het menuobject in een aparte stap toewijzen aan de eigenschap `submenu` ervan:

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

Menuopdrachten maken

Adobe AIR 1.0 of hoger

Als u een menuopdracht wilt maken, voegt u een NativeMenuItem-object toe aan een menu. Voeg er vervolgens een gebeurtenislistener aan toe die verwijst naar de functie die de menuopdracht implementeert:

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

U kunt luisteren naar de gebeurtenis `select` op het opdrachtitem zelf (zoals weergegeven in het voorbeeld), of u kunt luisteren naar de gebeurtenis `select` op een bovenliggend menuobject.

Opmerking: *Menu-items die submenu's en scheidingslijnen vertegenwoordigen, verzenden de gebeurtenis `select` niet en kunnen dus niet als opdracht worden gebruikt.*

Menuscheidingslijnen maken

Adobe AIR 1.0 of hoger

Als u een scheidingslijn wilt maken, maakt u een NativeMenuItem-object. Stel de parameter `isSeparator` in de constructor in op `true`. Vervolgens voegt u het item voor de scheidingslijn op de gewenste locatie toe aan het menu.

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

Een eventueel label dat voor de scheidingslijn wordt opgegeven, wordt niet weergegeven.

Informatie over contextmenu's in HTML (AIR)

Adobe AIR 1.0 of hoger

In HTML-inhoud die wordt weergegeven met het HTMLLoader-object, kunt u met de gebeurtenis `contextmenu` een contextmenu weergeven. Standaard wordt een contextmenu automatisch weergegeven wanneer de gebruiker de gebeurtenis contextmenu voor geselecteerde tekst activeert (door met de rechtermuisknop te klikken of de Command-toets ingedrukt te houden tijdens het klikken). Om te voorkomen dat het standaardmenu wordt geopend, luistert u naar de gebeurtenis `contextmenu` en roept u de methode `preventDefault()` van het gebeurtenisobject op:

```
function showContextMenu(event){
    event.preventDefault();
}
```

U kunt dan een aangepast contextmenu weergeven met behulp van DHTML-technieken of een native contextmenu van AIR weergeven. In het volgende voorbeeld wordt een native contextmenu weergegeven door het oproepen van de menumethode `display()` als reactie op de HTML-gebeurtenis `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>
```

Native pop-upmenu's (AIR) weergeven

Adobe AIR 1.0 of hoger

U kunt elk willekeurig NativeMenu-object op elk gewenst moment en een willekeurige locatie boven een venster weergeven door de menumethode `display()` op te roepen. Voor deze methode is een verwijzing naar het werkgebied nodig. Alleen inhoud in de toepassingssandbox kan dus een menu weergeven als pop-up.

Met de volgende methode wordt als reactie op een muisklik het menu weergegeven dat wordt gedefinieerd door een NativeMenu-object met de naam `popupMenu`:

```
 private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

***Opmerking:** Het menu hoeft niet te worden weergegeven als directe reactie op een gebeurtenis. Iedere methode kan de functie `display()` oproepen.*

Menugebeurtenissen afhandelen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een menu verzendt gebeurtenissen wanneer de gebruiker het menu of een menu-item selecteert.

Overzicht van de gebeurtenissen voor menuklassen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voeg gebeurtenislisteners aan menu's of individuele items toe om menugebeurtenissen af te handelen.

Object	Verzonden gebeurtenissen
NativeMenu (AIR)	Event.PREPARING (Adobe AIR 2.6 of hoger) Event.DISPLAYING Event.SELECT (doorgegeven door onderliggende items en submenu's)
NativeMenuItem (AIR)	Event.PREPARING (Adobe AIR 2.6 of hoger) Event.SELECT Event.DISPLAYING (doorgegeven door het bovenliggende menu)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT (AIR)

Select-menugebeurtenissen

Adobe AIR 1.0 of hoger

Om een klik op een menu-item af te handelen, voegt u een gebeurtenislistener voor de gebeurtenis `select` toe aan het NativeMenuItem-object:

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

Omdat `select`-gebeurtenissen worden doorgegeven naar de bevattende menu's, kunt u ook luisteren naar select-gebeurtenissen in een bovenliggend menu. Wanneer u luistert op menuniveau, kunt u de eigenschap `target` van het gebeurtenisobject gebruiken om te controleren welke menuopdracht is geselecteerd. In het volgende voorbeeld wordt het label van de geselecteerde opdracht getraceerd:

```
 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");
}
```

Als u de klasse ContextMenuItem gebruikt, kunt u luisteren naar de gebeurtenis `select` of `menuItemSelect`. De gebeurtenis `menuItemSelect` geeft u aanvullende informatie over het object waarvan het contextmenu eigendom is, maar wordt niet doorgegeven naar de bevattende menu's.

Displaying-menugebeurtenissen

Adobe AIR 1.0 of hoger

Om het openen van een menu af te handelen, kunt u een listener toevoegen voor de gebeurtenis `displaying` die wordt verzonden voordat een menu wordt weergegeven. U kunt de gebeurtenis displaying gebruiken om het menu te updaten, bijvoorbeeld door items toe te voegen of te verwijderen, of door de ingeschakelde of aangevinkte status van afzonderlijke items te updaten. U kunt ook luisteren naar de gebeurtenis `menuSelect` in een ContextMenu.

In AIR 2.6 of hoger kunt u met de `preparing`-gebeurtenis een menu bijwerken als reactie op het weergeven van een menu of het selecteren van een item met een sneltoets.

Voorbeeld van native menu: venster- en toepassingsmenu (AIR)

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u het menu maakt dat wordt weergegeven in "Native menustructuur (AIR)" op pagina 670.

Dit menu is zo ontworpen dat het zowel werkt in Windows (waarvoor alleen venstermenu's worden ondersteund) als in Mac OS X (waarvoor alleen toepassingsmenu's worden ondersteund). Om dit onderscheid kunnen maken, controleert de constructor van de klasse MenuExample de statische `supportsMenu`-eigenschappen van de klassen NativeWindow en NativeApplication. Als `NativeWindow.supportsMenu` de waarde `true` heeft, maakt de constructor een NativeMenu-object voor het venster, waarna de submenu's File en Edit worden gemaakt en toegevoegd. Als `NativeApplication.supportsMenu` de waarde `true` heeft, maakt de constructor de menu's File en Edit, waarna deze worden toegevoegd aan het bestaande menu dat wordt geleverd door het besturingssysteem Mac OS X.

Dit voorbeeld illustreert ook de afhandeling van menugebeurtenissen. De gebeurtenis `select` wordt zowel op item- als op menuniveau afgehandeld. Ieder menu in de keten, van het menu dat het geselecteerde item bevat tot het hoofdmenu, reageert op de gebeurtenis `select`. De gebeurtenis `displaying` wordt gebruikt met het menu met recent geopende bestanden. Vlak voordat het menu wordt geopend, worden de items in het menu vernieuwd aan de hand van de array met recent geopende documenten (die in dit voorbeeld niet verandert). Hoewel het in dit voorbeeld niet wordt geïllustreerd, kunt u ook luisteren naar `displaying`-gebeurtenissen voor afzonderlijke items.

```
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;
        }
    }
}
```

Hoofdstuk 37: Taakbalkpictogrammen in AIR

Adobe AIR 1.0 of hoger

Veel besturingssystemen hebben een taakbalk, zoals het dock van Mac OS X, dat pictogrammen (in Mac OS 'symbolen') kan bevatten die toepassingen vertegenwoordigen. Adobe® AIR® biedt een interface voor interactie met het taakbalkpictogram van een toepassing via de eigenschap `NativeApplication.nativeApplication.icon`.

- Werken met systeemvak- en dock-pictogrammen (Flex)
- Werken met systeemvak- en Dock-pictogrammen (Flash)

Meer Help-onderwerpen

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

Informatie over taakbalkpictogrammen

Adobe AIR 1.0 of hoger

AIR maakt automatisch het object `NativeApplication.nativeApplication.icon`. Het objecttype is DockIcon of SystemTrayIcon, afhankelijk van het besturingssysteem. U kunt bepalen welke van deze InteractiveIcon-subklassen AIR in het huidige besturingssysteem ondersteunt met behulp van de eigenschappen `NativeApplication.supportsDockIcon` en `NativeApplication.supportsSystemTrayIcon`. De basisklasse InteractiveIcon biedt de eigenschappen `width`, `height` en `bitmaps`, die u kunt gebruiken om de afbeelding voor het pictogram te wijzigen. Als u echter eigenschappen die specifiek zijn voor DockIcon of SystemTrayIcon in het verkeerde besturingssysteem opvraagt, wordt er een runtimefout gegenereerd.

Als u de afbeelding die voor een pictogram wordt gebruikt, wilt instellen of wijzigen, maakt u een array met een of meer afbeeldingen en wijst u deze toe aan de eigenschap `NativeApplication.nativeApplication.icon.bitmaps`. De grootte van taakbalkpictogrammen kan per besturingssysteem verschillen. Als u wilt voorkomen dat de kwaliteit van de pictogramafbeelding vermindert door het wijzigen van het formaat, voegt u afbeeldingen in verschillende formaten toe aan de array `bitmaps`. Als u meer dan één afbeelding aanbiedt, selecteert AIR automatisch het formaat dat het dichtst bij de huidige weergavegrootte van het taakbalkpictogram komt, en wordt het formaat van de afbeelding alleen aangepast als dat noodzakelijk is. In het volgende voorbeeld ziet u hoe u de afbeelding voor een taakbalkpictogram instelt met twee afbeeldingen:

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

Als u de pictogramafbeelding wilt wijzigen, wijst u een array met de nieuwe afbeelding(en) toe aan de eigenschap `bitmaps`. U kunt animatie aan het pictogram toevoegen door de afbeelding te wijzigen als reactie op de gebeurtenis `enterFrame` of `timer`.

Als u het pictogram uit het systeemvak in Windows of Linux wilt verwijderen, of de standaardweergave van het pictogram wilt herstellen in Mac OS X, stelt u `bitmaps` in op een lege array:

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

Dock-pictogrammen

Adobe AIR 1.0 of hoger

AIR ondersteunt pictogrammen in het dock wanneer `NativeApplication.supportsDockIcon` is ingesteld op `true`. De eigenschap `NativeApplication.nativeApplication.icon` vertegenwoordigt het toepassingspictogram in het dock (geen vensterpictogram in het dock).

Opmerking: *AIR biedt geen ondersteuning voor veranderende vensterpictogrammen in het dock in Mac OS X. Ook zijn wijzigingen in het dock-pictogram voor een toepassing alleen van kracht op het moment dat de toepassing wordt uitgevoerd en krijgt het pictogram weer zijn normale uiterlijk wanneer de toepassing wordt afgesloten.*

Menu's van pictogrammen in het dock

Adobe AIR 1.0 of hoger

U kunt opdrachten toevoegen aan het standaardmenu van een dock door een NativeMenu-object met deze opdrachten te maken en dit object vervolgens toe te wijzen aan de eigenschap `NativeApplication.nativeApplication.icon.menu`. De items in het menu worden boven de standaarditems in het dock-menu weergegeven.

Het pictogram in het dock laten bewegen

Adobe AIR 1.0 of hoger

U kunt een pictogram in het dock laten bewegen door de methode `NativeApplication.nativeApplication.icon.bounce()` op te roepen. Als u de parameter `bounce() priority` instelt op informational (informatief), beweegt het pictogram één keer. Als u deze parameter instelt op critical (kritiek), beweegt het pictogram totdat de gebruiker de toepassing activeert. Constanten voor de parameter `priority` worden gedefineerd in de klasse NotificationType.

Opmerking: *Het pictogram beweegt niet als de toepassing al actief is.*

Gebeurtenissen van pictogrammen in het dock

Adobe AIR 1.0 of hoger

Wanneer op een dock-pictogram wordt geklikt, verzendt het NativeApplication-object de gebeurtenis `invoke`. Als de toepassing nog niet wordt uitgevoerd, wordt deze door het systeem gestart. Als de toepassing wel wordt uitgevoerd, wordt de gebeurtenis `invoke` verzonden naar de actieve toepassingsinstantie.

Systeemvakpictogrammen

Adobe AIR 1.0 of hoger

AIR ondersteunt systeemvakpictogrammen wanneer `NativeApplication.supportsSystemTrayIcontrue` is. Dat is gewoonlijk het geval in Windows en de meeste Linux-distributies. In Windows en Linux worden systeemvakpictogrammen weergegeven in het systeemvak van de taakbalk. Er wordt standaard geen enkel pictogram weergegeven. Als u een pictogram wilt weergeven, wijst u een array met BitmapData-objecten toe aan de eigenschap `bitmaps` van het pictogram. Als u de pictogramafbeelding wilt wijzigen, wijst u een array met de nieuwe afbeeldingen toe aan de eigenschap `bitmaps`. Als u het pictogram wilt verwijderen, stelt u `bitmaps` in op `null`.

Menu's van systeemvakpictogrammen

Adobe AIR 1.0 of hoger

U kunt een menu aan een systeemvakpictogram toevoegen door een NativeMenu-object te maken en dit object vervolgens aan de eigenschap `NativeApplication.nativeApplication.icon.menu` toe te wijzen (er is geen standaardmenu beschikbaar in het besturingssysteem). U kunt het menu van het systeemvakpictogram openen door met de rechtermuisknop op het pictogram te klikken.

Knopinfo van systeemvakpictogrammen

Adobe AIR 1.0 of hoger

Voeg knopinfo toe aan een pictogram door de eigenschap voor knopinfo in te stellen:

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

Gebeurtenissen van systeemvakpictogrammen

Adobe AIR 1.0 of hoger

Het object SystemTrayIcon waarnaar wordt verwezen met de eigenschap NativeApplication.nativeApplication.icon, verzendt een ScreenMouseEvent voor de gebeurtenissen `click`, `mouseDown`, `mouseUp`, `rightClick`, `rightMouseDown` en `rightMouseUp`. U kunt deze gebeurtenissen samen met een pictogrammenu gebruiken om gebruikers in staat te stellen met uw toepassing te interageren wanneer deze geen zichtbare vensters heeft.

Voorbeeld: een toepassing maken zonder vensters

Adobe AIR 1.0 of hoger

In het volgende voorbeeld wordt een AIR-toepassing gemaakt met een systeemvakpictogram maar zonder zichtbare vensters. (De eigenschap `visible` van de toepassing mag niet zijn ingesteld op `true` in de toepassingsdescriptor. Als dit wel het geval is, zal het venster zichtbaar zijn wanneer de toepassing wordt opgestart.)

```
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];
        }
    }
}
```

Opmerking: *Als de Flex-component WindowedApplication wordt gebruikt, moet u het kenmerk* `visible` *van de tag WindowedApplication instellen op* `false`*. Dit kenmerk heeft voorrang op de instellingen in de toepassingsdescriptor.*

Opmerking: *In het voorbeeld wordt aangenomen dat er afbeeldingsbestanden aanwezig zijn met de naam* `AIRApp_16.png` *en* `AIRApp_128.png` *in de submap* `icons` *van de toepassing. (In de SDK van AIR vindt u voorbeeldpictogrambestanden, die u naar uw eigen projectmap kunt kopiëren.)*

Taakbalkpictogrammen en -knoppen voor vensters

Adobe AIR 1.0 of hoger

In een speciaal gebied van de taakbalk of het dock worden gewoonlijk pictogrammen of knoppen weergegeven die vensters vertegenwoordigen en waarmee gebruikers gemakkelijk toegang kunnen krijgen tot vensters die zich op de achtergrond bevinden of zijn geminimaliseerd. In het dock van Mac OS X wordt een pictogram weergegeven voor een toepassing, maar ook een pictogram voor elk geminimaliseerd venster. Op de taakbalken van Microsoft Windows en Linux wordt een knop weergegeven met het programmapictogram en de titel van elk gewoon venster in de toepassing.

De taakbalkknop voor een venster markeren

Adobe AIR 1.0 of hoger

Wanneer een venster zich op de achtergrond bevindt, kunt u de gebruiker waarschuwen wanneer betreffende dat venster een gebeurtenis plaatsvindt die van belang is voor de gebruiker. In Mac OS X kunt u de gebruiker waarschuwen door het toepassingspictogram in het dock te laten bewegen (zoals is beschreven in "Het pictogram in het dock laten bewegen" op pagina 684). In Windows en Linux kunt u de taakbalkknop van het venster markeren door de methode `notifyUser()` van het NativeWindow-exemplaar aan te roepen. De parameter `type` die aan de methode wordt doorgegeven, bepaalt de urgentie van de waarschuwing:

- `NotificationType.CRITICAL`: het pictogram van het venster knippert totdat de gebruiker het venster op de voorgrond plaatst.
- `NotificationType.INFORMATIONAL`: het pictogram van het venster wordt gemarkeerd door een kleurwijziging.

  Opmerking: *In Linux wordt alleen de melding van het informatieve type ondersteund. Wanneer u een waarde van een van deze typen aan de functie* `notifyUser()` *doorgeeft, leidt dit tot hetzelfde effect.*

  Met de volgende instructie zorgt u dat de taakbalkknop van een venster wordt gemarkeerd:

  ```
  stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
  ```

  De methode `NativeWindow.notifyUser()` heeft alleen effect als u deze methode oproept in een besturingssysteem dat waarschuwingen op vensterniveau ondersteunt. Gebruik de eigenschap `NativeWindow.supportsNotification` om te controleren of vensterwaarschuwingen worden ondersteund.

Vensters zonder taakbalkknoppen of -pictogrammen maken

Adobe AIR 1.0 of hoger

In Windows worden vensters van het type *utility* of *lightweight* niet op de taakbalk weergegeven. Ook onzichtbare vensters verschijnen niet op de taakbalk.

Omdat het beginvenster noodzakelijkerwijs van het type *normal* is, moet u dit beginvenster sluiten of zorgen dat het onzichtbaar blijft als u een toepassing wilt maken waarvan de vensters niet als knop op de taakbalk worden weergegeven. Als u alle vensters van uw toepassing wilt sluiten zonder de toepassing af te sluiten, stelt u de eigenschap `autoExit` van het object NativeApplication in op `false` voordat u het laatste venster sluit. Als u eenvoudig wilt voorkomen dat het beginvenster ooit zichtbaar wordt, voegt u `<visible>false</visible>` toe aan het element `<initialWindow>` van het descriptorbestand van de toepassing. (Stel de eigenschap `visible` niet in op `true` en roep ook niet de methode `activate()` van het venster op.)

Stel voor nieuwe vensters die door de toepassing worden geopend, de eigenschap `type` van het object NativeWindowInitOption dat wordt doorgegeven aan de vensterconstructor, in op `NativeWindowType.UTILITY` of `NativeWindowType.LIGHTWEIGHT`.

In Mac OS X worden geminimaliseerde vensters weergegeven op de dock-taakbalk. U kunt voorkomen dat het pictogram van een geminimaliseerd venster wordt weergegeven door het venster te verbergen in plaats van te minimaliseren. In het volgende voorbeeld wordt geluisterd naar een gebeurtenis die `nativeWindowDisplayState` verandert en wordt deze gebeurtenis geannuleerd als het venster wordt geminimaliseerd. In plaats daarvan stelt de handler de eigenschap `visible` van het venster in op `false`:

```
private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

Als een venster wordt geminimaliseerd in het dock van Mac OS X wanneer u de eigenschap `visible` op `false` instelt, wordt het pictogram niet uit het dock verwijderd. Een gebruiker kan in dat geval nog steeds op het pictogram klikken om het venster opnieuw weer te geven.

Hoofdstuk 38: Werken met het bestandssysteem

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash® Player biedt basismogelijkheden voor het lezen en schrijven van bestanden, via de FileReference-klasse. Voor veiligheidsredenen moet de gebruiker, wanneer deze in Flash Player wordt uitgevoerd, altijd toestemming verlenen voordat u een bestand kunt lezen of schrijven.

Adobe® AIR® biedt een completere toegang tot het bestandssysteem van de hostcomputer dan met Flash Player. Met de API voor het AIR-bestandssysteem kunt u mappen en bestanden maken, openen en beheren, gegevens naar bestanden schrijven, enz.

Meer Help-onderwerpen

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

De FileReference-klasse gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een FileReference-object vertegenwoordigt een gegevensbestand op een client- of servercomputer. Met de methoden van de klasse FileReference kan uw toepassing gegevensbestanden lokaal laden en opslaan, en bestandsgegevens van en naar externe servers overbrengen.

De klasse FileReference biedt twee verschillende benaderingen om gegevensbestanden te laden, over te brengen en op te slaan. Sinds de introductie van de klasse FileReference bevat deze klasse de methode `browse()`, de methode `upload()` en de methode `download()`. Gebruik de methode `browse()` om de gebruiker een bestand te laten selecteren. Gebruik de methode `upload()` om bestandsgegevens naar een externe server over te brengen. Gebruik de methode `download()` om die gegevens weer van de server op te halen en ze in een lokaal bestand op te slaan. Vanaf Flash Player 10 en Adobe AIR 1.5 bevat de klasse FileReference eveneens de methoden `load()` en `save()`. Met de methoden `load()` en `save()` hebt u rechtstreeks toegang tot lokale bestanden en kunt u deze eveneens opslaan. U gebruikt deze methoden op dezelfde manier als de gelijknamige methoden in de klassen URLLoader en Loader.

Opmerking: *De File-klasse, die de FileReference-klasse bevat, en de FileStream-klasse bieden extra functies voor het werken met bestanden en het lokale bestandssysteem. De File- en FileStream-klassen worden alleen ondersteund in AIR en niet in Flash Player.*

FileReference, klasse

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elk FileReference-object vertegenwoordigt één gegevensbestand op de lokale machine. De eigenschappen van de FileReference-klasse bevat informatie over de grootte, type, naam, bestandsnaam, bestandsextensie, auteur, aanmaakdatum en aanpassingsdatum van het bestand.

Opmerking: *De eigenschap* `creator` *wordt alleen in Mac OS ondersteund. Op alle andere platformen wordt* `null` *geretourneerd.*

Opmerking: *De eigenschap* `extension` *wordt alleen in Adobe AIR ondersteund.*

U kunt op twee manieren een instantie van de klasse FileReference maken:

- Gebruik de operator `new` zoals in de volgende code wordt geïllustreerd:

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- Roep de methode `FileReferenceList.browse()` op, waarmee een dialoogvenster wordt geopend waarin de gebruiker wordt gevraagd een of meer bestanden te selecteren om te uploaden. Vervolgens wordt een array met FileReference-objecten gemaakt als de gebruiker een of meer bestanden heeft geselecteerd.

Wanneer u een FileReference-object hebt gemaakt, kunt u het volgende doen:

- Roep de methode `FileReference.browse()` op, waarmee een dialoogvenster wordt geopend waarin de gebruiker wordt gevraagd één bestand in het lokale bestandssysteem te selecteren. Dit gebeurt gewoonlijk vóór een daaropvolgende aanroep van de methode `FileReference.upload()` of de methode `FileReference.load()`. Roep de methode `FileReference.upload()` aan om het bestand naar een externe server te uploaden. Roep de methode `FileReference.load()` aan om een lokaal bestand te openen.
- Roep de methode `FileReference.download()` aan. De `download()`-methode opent een dialoogvenster zodat de gebruiker een locatie kan selecteren waar een nieuw bestand wordt opgeslagen. Daarna worden de gegevens van de server gedownload en in het nieuwe bestand opgeslagen.
- Roep de methode `FileReference.load()` aan. Deze methode begint gegevens te laden uit een bestand dat eerder is geselecteerd met de methode `browse()`. De methode `load()` kan pas worden aangeroepen wanneer de bewerking `browse()` is voltooid (dus wanneer de gebruiker een bestand heeft geselecteerd).
- Roep de methode `FileReference.save()` aan. Deze methode opent een dialoogvenster waarin de gebruiker wordt gevraagd een enkele bestandslocatie op het lokale bestandssysteem te selecteren. Daarna worden de gegevens op de opgegeven locatie opgeslagen.

Opmerking: *U kunt maar één* `browse()`*-,* `download()`*- of* `save()`*-actie tegelijk uitvoeren, aangezien er maar één dialoogvenster tegelijk kan zijn geopend.*

De eigenschappen van het FileReference-object, zoals `name`, `size` en `modificationDate`, worden pas in een van de volgende omstandigheden gedefinieerd:

- De methode `FileReference.browse()` of `FileReferenceList.browse()` is opgeroepen en de gebruiker heeft een bestand geselecteerd in het dialoogvenster.
- De methode `FileReference.download()` is opgeroepen en de gebruiker heeft een nieuwe bestandslocatie in het dialoogvenster opgegeven.

Opmerking: *Wanneer een bestand wordt gedownload, wordt alleen de eigenschap* `FileReference.name` *met gegevens gevuld voordat de download is voltooid. Nadat het bestand is gedownload, zijn alle andere eigenschappen beschikbaar.*

Terwijl oproepen van de methoden `FileReference.browse()`, `FileReferenceList.browse()`, `FileReference.download()`, `FileReference.load()` of `FileReference.save()` worden uitgevoerd, gaan de meeste spelers door met het afspelen van SWF-bestanden, waaronder het verzenden van gebeurtenissen en het uitvoeren van code.

Voor het uploaden en downloaden heeft een SWF-bestand alleen toegang tot bestanden binnen het eigen domein, inclusief alle domeinen die door een beleidsbestand zijn opgegeven. U moet een beleidsbestand plaatsen op de server die het bestand bevat als de server zich niet in hetzelfde domein bevindt als het SWF-bestand dat de upload of download heeft gestart.

Zie FileReference.

Gegevens uit bestanden laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `FileReference.load()` kunt u gegevens uit een lokaal bestand in het geheugen laden.

Opmerking: *De code moet eerst de methode `FileReference.browse()` aanroepen opdat de gebruiker het te laden bestand kan selecteren. Deze beperking is niet van toepassing op inhoud die wordt uitgevoerd in Adobe AIR in de beveiligingssandbox van de toepassing.*

De methode `FileReference.load()` keert onmiddellijk terug nadat deze is aangeroepen, maar de te laden gegevens zijn niet onmiddellijk beschikbaar. Het FileReference-object verzendt gebeurtenissen om listenermethoden te activeren bij elke stap van het laadproces.

Het FileReference-object verzendt tijdens het laadproces de volgende gebeurtenissen.

- `open`-gebeurtenis (`Event.OPEN`): Wordt verzonden wanneer de laadbewerking start.
- `progress`-gebeurtenis (`ProgressEvent.PROGRESS`): Wordt regelmatig verzonden naarmate gegevensbytes uit het bestand worden gelezen.
- `complete`-gebeurtenis (`Event.COMPLETE`): Wordt verzonden wanneer de laadbewerking met succes is voltooid.
- `ioError`-gebeurtenis (`IOErrorEvent.IO_ERROR`): Wordt verzonden als het laadproces is mislukt als gevolg van een invoer-/uitvoerfout tijdens het openen of lezen van gegevens uit het bestand.

Zodra het FileReference-object de volledige gebeurtenis heeft verzonden, kunnen de geladen gegevens worden benaderd als een ByteArray in de eigenschap `data` van het FileReference-object.

In het volgende voorbeeld ziet u hoe de gebruiker wordt gevraagd een bestand te selecteren en hoe de gegevens vervolgens uit dat bestand in het geheugen worden geladen:

```
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.");
        }
    }
}
```

In de voorbeeldcode wordt eerst een FileReference-object met de naam `fileRef` gemaakt, en daarna wordt de methode `browse()` van dit object aangeroepen. De `browse()`-methode opent een dialoogvenster die de gebruiker vraagt om een bestand te selecteren. Als een bestand is geselecteerd wordt de methode `onFileSelected()` door de code aangeroepen. Deze methode voegt listeners voor de gebeurtenissen `progress` en `complete` toe en roept vervolgens de methode `load()` van het FileReference-object aan. De andere handlermethoden in het voorbeeld genereren alleen berichten om te rapporteren over de voortgang van de laadbewerking. Wanneer het laden is voltooid, geeft de toepassing de inhoud van het geladen bestand weer met behulp van de methode `trace()`.

In Adobe AIR, biedt de FileStream-klasse een aanvullende functionaliteit voor het lezen van gegevens van een lokaal bestand. Zie "Lezen van en schrijven naar bestanden" op pagina 728.

Gegevens opslaan in lokale bestanden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `FileReference.save()` kunt u gegevens opslaan in een lokaal bestand. Deze methode begint met het openen van een dialoogvenster waarin de gebruiker een nieuwe bestandsnaam en de locatie kan opgeven waar het bestand moet worden opgeslagen. Nadat de gebruiker de bestandsnaam en -locatie heeft geselecteerd, worden de gegevens naar het nieuwe bestand geschreven. Wanneer het bestand is opgeslagen, worden de eigenschappen van het object FileReference gevuld met de eigenschappen van het lokale bestand.

Opmerking: *Uw code kan alleen de `FileReference.save()`-methode aanroepen als reactie op een gebeurtenis die door een gebruiker is gestart, zoals een muisklik of een actie op het toetsenbord. Anders wordt een fout gegenereerd. Deze beperking is niet van toepassing op actieve inhoud in Adobe AIR in de beveiligingssandbox van de toepassing.*

De methode `FileReference.save()` retourneert onmiddellijk na de oproep. Het FileReference-object verzendt vervolgens gebeurtenissen om listenermethoden te activeren bij elke stap van het opslagproces.

Het FileReference-object verzendt de volgende gebeurtenissen tijdens het opslagproces:

- `select`-gebeurtenis (`Event.SELECT`): Wordt verzonden wanneer de gebruiker de locatie en bestandsnaam opgeeft voor het nieuwe bestand dat moet worden opgeslagen.
- `cancel`-gebeurtenis (`Event.CANCEL`): Wordt verzonden wanneer de gebruiker in het dialoogvenster op Annuleren klikt.
- `open`-gebeurtenis (`Event.OPEN`): Wordt verzonden wanneer de opslagbewerking start.
- `progress`-gebeurtenis (`ProgressEvent.PROGRESS`): Wordt regelmatig verzonden naarmate gegevensbytes in het bestand worden opgeslagen.
- `complete`-gebeurtenis (`Event.COMPLETE`): Wordt verzonden wanneer de opslagbewerking met succes is voltooid.
- `ioError`-gebeurtenis (`IOErrorEvent.IO_ERROR`): Wordt verzonden als het opslagproces mislukt als gevolg van een invoer-/uitvoerfout tijdens het opslaan van gegevens in het bestand.

Het type object dat in de parameter `data` van de methode `FileReference.save()` wordt doorgegeven, bepaalt hoe de gegevens naar het bestand worden geschreven:

- Als het een String-waarde is, wordt het bestand opgeslagen als een tekstbestand met UTF-8-codering.
- Als het een XML-object is, worden de gegevens weggeschreven naar een bestand in de XML-indeling waarbij alle opmaak bewaard blijft.
- Als het een ByteArray-object is, wordt de inhoud rechtstreeks en zonder conversie naar het bestand geschreven.
- Als het een ander type object is, roept de methode `FileReference.save()` de methode `toString()` van het object aan en wordt de resulterende String-waarde opgeslagen in een UTF-8-tekstbestand. Als de methode `toString()` van het object niet kan worden aangeroepen, wordt een fout gegenereerd.

Als de parameter `data` de waarde `null` heeft, wordt een fout gegenereerd.

De volgende code is een uitbreiding op het vorige voorbeeld van de methode `FileReference.load()`. Nadat de gegevens uit het bestand zijn gelezen, wordt de gebruiker in het voorbeeld naar een bestandsnaam gevraagd en worden de gegevens in een nieuw bestand opgeslagen:

```
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.");
        }
    }
}
```

Wanneer alle gegevens uit het bestand geladen zijn, wordt de methode `onComplete()` door de code aangeroepen. De methode `onComplete()` verwijdert de listeners voor de laadgebeurtenissen en roept vervolgens de methode `saveFile()` op. De methode `FileReference.save()` wordt door de methode `saveFile()` aangeroepen. Er wordt door de methode `FileReference.save()` een nieuw dialoogvenster geopend waarin de gebruiker een nieuwe bestandsnaam en -locatie kan selecteren om het bestand op te slaan. De overige gebeurtenislistenermethoden volgen de voortgang van het opslagproces tot het is voltooid.

In Adobe AIR, biedt de FileStream-klasse extra functionaliteit voor het schrijven van gegevens naar een lokaal bestand. Zie “Lezen van en schrijven naar bestanden” op pagina 728.

Bestanden uploaden naar een server

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u bestanden wilt uploaden naar een server, roept u eerst de methode `browse()` aan, waarmee de gebruiker een of meer bestanden kan selecteren. Wanneer vervolgens de methode `FileReference.upload()` wordt aangeroepen, wordt het geselecteerde bestand overgebracht naar de server. Als de gebruiker meerdere bestanden selecteert met de methode `FileReferenceList.browse()`, maakt Flash Player een array met de geselecteerde bestanden, met de naam `FileReferenceList.fileList`. U kunt vervolgens elk bestand afzonderlijk uploaden met de methode `FileReference.upload()`.

Opmerking: *Met de methode `FileReference.browse()` kan slechts één bestand worden geüpload. Als u wilt dat de gebruiker meerdere bestanden kan uploaden, moet u de methode `FileReferenceList.browse()` gebruiken.*

In het dialoogvenster met de bestandskiezer van het besturingssysteem kunnen gebruikers standaard elk bestandstype op de lokale computer selecteren. Ontwikkelaars kunnen een of meer aangepaste bestandstypefilters opgeven door de klasse FileFilter te gebruiken en een array met bestandsfilterinstanties door te geven aan de methode `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);
```

Nadat de gebruiker de bestanden heeft geselecteerd en op de knop Openen in de bestandskiezer heeft geklikt, wordt de gebeurtenis `Event.SELECT` verzonden. Als de methode `FileReference.browse()` wordt gebruikt om een bestand voor het uploaden te selecteren, wordt het bestand door de volgende code naar een webserver verzonden:

```
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");
}
```

U kunt gegevens naar de server verzenden met de methode `FileReference.upload()` door gebruik te maken van de eigenschappen `URLRequest.method` en `URLRequest.data` om variabelen te verzenden met de methode `POST` of `GET`.

Wanneer u probeert om een bestand te laden met de `FileReference.upload()`-methode, worden de volgende gebeurtenissen verzonden:

- `open`-gebeurtenis (`Event.OPEN`): Wordt verzonden wanneer de laadbewerking start.
- `progress`-gebeurtenis (`ProgressEvent.PROGRESS`): Wordt regelmatig verzonden naarmate gegevensbytes uit het bestand worden geladen.
- `complete`-gebeurtenis (`Event.COMPLETE`): Wordt verzonden wanneer de laadbewerking met succes is voltooid.
- `httpStatus`-gebeurtenis (`HTTPStatusEvent.HTTP_STATUS`): Wordt verzonden wanneer het laadproces mislukt door een HTTP-fout.
- `httpResponseStatus`-gebeurtenis (`HTTPStatusEvent.HTTP_RESPONSE_STATUS`): Wordt verzonden als een aanroep van de methode `upload()` of `uploadUnencoded()` via HTTP toegang tot gegevens probeert te krijgen, en Adobe AIR de statuscode voor de aanvraag kan detecteren en retourneren.
- `securityError`-gebeurtenis (`SecurityErrorEvent.SECURITY_ERROR`Verzonden wanneer een oplaadactie mislukt als gevolg van een schending van de beveiliging.
- `uploadCompleteData`-gebeurtenis (`DataEvent.UPLOAD_COMPLETE_DATA`): Wordt verzonden nadat gegevens van de server zijn ontvangen nadat het uploaden is voltooid.
- `ioError`-gebeurtenis (`IOErrorEvent.IO_ERROR`): Wordt verzonden als het uploadproces mislukt door een van de volgende oorzaken:
  - Er is een invoer-/uitvoerfout opgetreden terwijl Flash Player het bestand las, schreef of verzond.
  - Het SWF-bestand probeert een bestand te uploaden dat verificatie vereist (zoals een gebruikersnaam en een wachtwoord). Tijdens het uploaden biedt Flash Player gebruikers niet de mogelijkheid een wachtwoord in te voeren.
  - De parameter `url` bevat een ongeldig protocol. De methode `FileReference.upload()` moet HTTP of HTTPS gebruiken.

Flash Player biedt geen volledige ondersteuning van servers die verificatie vereisen. Alleen SWF-bestanden die in een browser worden uitgevoerd, waarbij de browserinsteekmodule of het Microsoft ActiveX®-besturingselement wordt gebruikt, kunnen een dialoogvenster weergeven waarin de gebruiker wordt gevraagd een gebruikersnaam en wachtwoord ter verificatie in te voeren. Bovendien is dit alleen mogelijk voor downloads. Bij uploaden met behulp van de insteekmodule of het ActiveX-besturingselement, of bij uploaden/downloaden met een zelfstandige of externe speler, mislukt de bestandsoverdracht.

Als u in ColdFusion een serverscript maakt dat een bestandsupload van Flash Player accepteert, kunt u code gebruiken zoals hier beschreven:

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#"
nameconflict="OVERWRITE" />
```

Deze ColdFusion-code uploadt het bestand dat door Flash Player is verzonden en slaat het op in dezelfde map als de ColdFusion-sjabloon. Als er een bestand met dezelfde naam aanwezig is, wordt dit overschreven. De voorgaande code bevat de minimale code die nodig is om een bestandsupload te accepteren. Dit script moet niet worden gebruikt in een productie-omgeving. U kunt het beste gegevensvalidatie toevoegen zodat gebruikers alleen geaccepteerde bestandstypen kunnen uploaden, zoals een afbeelding in plaats van een potentieel gevaarlijk serverscript.

De volgende code bevat een voorbeeld van een bestandsupload met PHP en bevat gegevensvalidatie. Het script beperkt het aantal geüploade bestanden in de uploadmap tot 10, controleert of het bestand kleiner is dan 200 kB en staat alleen toe dat JPEG-, GIF- of PNG-bestanden worden geüpload naar en opgeslagen op het bestandssysteem.

```
<?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;
    }
}
?>
```

U kunt aanvullende variabelen doorgeven aan het uploadscript met de aanvraagmethode POST of GET. Als u aanvullende POST-variabelen naar het uploadscript wilt verzenden, kunt u de volgende code gebruiken:

```
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");
}
```

In het voorgaande voorbeeld wordt een URLVariables-object gemaakt dat u doorgeeft aan het externe serverscript. In vorige versies van ActionScript kon u variabelen doorgeven aan het serveruploadscript door waarden door te geven in de queryreeks. In ActionScript 3.0 kunt u variabelen aan het externe script doorgeven met een object URLRequest, zodat u gegevens kunt doorgeven met de methode `POST` of `GET`. Hiermee kunnen grotere hoeveelheden gegevens gemakkelijker worden doorgegeven. Als u wilt opgeven of de variabelen worden doorgegeven met de aanvraagmethode `GET` of `POST`, stelt u de eigenschap `URLRequest.method` in op respectievelijk `URLRequestMethod.GET` of `URLRequestMethod.POST`.

In ActionScript 3.0 kunt u de standaardveldnaam voor uploadbestanden `Filedata` overschrijven door een tweede parameter door te geven aan de methode `upload()`, zoals in het volgende voorbeeld wordt getoond (waarin de standaardwaarde `Filedata` wordt vervangen door `Custom1`).

De Flash Player probeert standaard geen testupload te verzenden, hoewel u deze standaard kan negeren door een waarde van `true` te passeren als de derde parameter van de `upload()`-methode. Het doel van de testupload is te controleren of de daadwerkelijke bestandsupload en de serververificatie, indien vereist, zal slagen.

Opmerking: *Momenteel vindt een testupload alleen plaats in Flash Player voor Windows.*

Het serverscript dat de bestandsupload afhandelt, moet een HTTP `POST`-aanvraag verwachten met de volgende elementen:

- `Content-Type` met de waarde `multipart/form-data`.
- `Content-Disposition` waarbij het kenmerk `name` is ingesteld op "`Filedata`" en het kenmerk `filename` is ingesteld op de naam van het oorspronkelijke bestand U kunt een aangepast kenmerk `name` opgeven door een waarde voor de parameter `uploadDataFieldName` door te geven in de methode `FileReference.upload()`.
- De binaire inhoud van het bestand.

Hier volgt een voorbeeld van een HTTP `POST`-aanvraag:

```
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,,,)
```

Met het volgende voorbeeld van een HTTP `POST`-aanvraag worden drie `POST`-variabelen verzonden: `api_sig`, `api_key` en `auth_token`, en wordt de aangepaste veldnaamwaarde voor uploadgegevens `"foto"` gebruikt:

```
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--
```

Bestanden downloaden vanaf een server

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt gebruikers bestanden laten downloaden vanaf een server met behulp van de methode `FileReference.download()`, die twee parameters gebruikt: `request` en `defaultFileName`. De eerste parameter is het object URLRequest dat de URL van het te downloaden bestand bevat. De tweede parameter is optioneel. Hiermee kunt u een standaardbestandsnaam opgeven die wordt weergegeven in het downloaddialoogvenster. Als u de tweede parameter weglaat, wordt `defaultFileName` gebruikt. Dit is de bestandsnaam uit de opgegeven URL.

Door de volgende code wordt een bestand met de naam index.xml gedownload uit dezelfde map als het SWF-document:

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

Als u de standaardnaam wilt instellen op currentnews.xml in plaats van index.xml, geeft u de parameter `defaultFileName` op, zoals in het volgende codefragment:

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

De naam van een bestand wijzigen kan nuttig zijn als de bestandsnaam op de server niet intuïtief is of door de server is gegenereerd. Het is ook raadzaam de parameter `defaultFileName` expliciet op te geven wanneer u een bestand downloadt met een serverscript in plaats van het bestand rechtstreeks te downloaden. U moet de parameter `defaultFileName` bijvoorbeeld opgeven als u een serverscript gebruikt dat specifieke bestanden downloadt op basis van doorgegeven URL-variabelen. Doet u dat niet, dan is de standaardnaam van het gedownloade bestand de naam van het serverscript.

U kunt gegevens naar de server verzenden met de methode `download()` door parameters aan de URL toe te voegen die het serverscript kan parseren. Het volgende ActionScript 3.0-fragment downloadt een document op basis waarvan parameters worden doorgegeven aan een ColdFusion-script:

```
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.");
            }
        }
    }
}
```

In de volgende code ziet u het ColdFusion-script, download.cfm, dat een van twee bestanden downloadt van de server, afhankelijk van de waarde van een URL-variabele:

```
<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>
```

FileReferenceList, klasse

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse FileReferenceList kan de gebruiker een of meer bestanden selecteren die naar een serverscript worden geüpload. De bestandsupload wordt afgehandeld door de methode `FileReference.upload()`, die moet worden aangeroepen voor elk bestand dat de gebruiker selecteert.

De volgende code maakt twee objecten FileFilter (`imageFilter` en `textFilter`) en geeft ze in een array door aan de methode `FileReferenceList.browse()`. Dit zorgt ervoor dat het bestandsdialoogvenster van het besturingssysteem twee mogelijke filters voor bestandstypen weergeeft.

```
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.");
}
```

Als u wilt dat de gebruiker een of meer bestanden kan selecteren en uploaden met de klasse FileReferenceList, staat dat gelijk aan het gebruik van `FileReference.browse()` voor het selecteren van bestanden. Met FileReferenceList kunnen echter meerdere bestanden worden geselecteerd. Voor het uploaden van meerdere bestanden moet u elk van de geselecteerde bestanden uploaden met behulp van `FileReference.upload()`, zoals in de volgende code:

```
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");
}
```

Aangezien de gebeurtenis `Event.COMPLETE` wordt toegevoegd aan elk afzonderlijk FileReference-object in de array, roept Flash Player de methode `completeHandler()` aan nadat elk afzonderlijk bestand is geüpload.

De API voor het AIR-bestandssysteem gebruiken

Adobe AIR 1.0 of hoger

De API voor het Adobe AIR-bestandssysteem bevat de volgende klassen:

- Bestand
- FileMode
- FileStream

Met de API voor het bestandssysteem kunt u het volgende (en meer) doen:

- Bestanden en mappen kopiëren, maken, verwijderen en verplaatsen.
- Informatie opvragen over bestanden en mappen
- Bestanden lezen en schrijven

Basisinformatie over AIR-bestanden

Adobe AIR 1.0 of hoger

Lees voor een snelle uitleg en codevoorbeelden van het werken met het bestandssysteem in AIR de snelstartartikelen in de Adobe Developer Connection:

- Een editor voor tekstbestanden maken (Flash)
- Een tekstbestandeditor bouwen (Flex)
- Een mapzoektoepassing bouwen (Flex)
- Vanaf een XML-voorkeurenbestand lezen en schrijven (Flex)
- Bestanden en gegevens comprimeren (Flex)

Adobe AIR biedt klassen die u kunt gebruiken om toegang te krijgen tot bestanden en mappen en om deze te maken en te beheren. Deze klassen, die zich bevinden in het pakket flash.filesystem, worden als volgt gebruikt:

File-klasse	Beschrijving
File	Het File-object vertegenwoordigt een pad naar een bestand of map. U gebruikt een File-object om een aanwijzer naar een bestand of map te maken, waarbij interactie met het bestand of de map wordt gestart.
FileMode	De klasse FileMode definieert tekenreeksconstanten die worden gebruikt in de parameter `fileMode` van de methoden `open()` en `openAsync()` van de klasse FileStream. De parameter `fileMode` van deze methoden bepaalt de mogelijkheden die beschikbaar zijn voor het FileStream-object als het bestand eenmaal is geopend. Het gaat hierbij om schrijven, lezen, toevoegen en bijwerken.
FileStream	Het FileStream-object wordt gebruikt om bestanden te openen voor lezen en schrijven. Als u een File-object hebt gemaakt dat een nieuw of bestaand bestand aanwijst, geeft u die aanwijzer door aan het FileStream-object zodat u het bestand kunt openen en gegevens kunt lezen of schrijven.

Bepaalde methoden in de klasse File hebben zowel een synchrone als een asynchrone versie:

- `File.copyTo()` en `File.copyToAsync()`
- `File.deleteDirectory()` en `File.deleteDirectoryAsync()`
- `File.deleteFile()` en `File.deleteFileAsync()`
- `File.getDirectoryListing()` en `File.getDirectoryListingAsync()`
- `File.moveTo()` en `File.moveToAsync()`
- `File.moveToTrash()` en `File.moveToTrashAsync()`

Ook werken FileStream-bewerkingen synchroon of asynchroon afhankelijk van de manier waarop het FileStream-object het bestand opent: door het oproepen van de methode `open()` of door het oproepen van de methode `openAsync()`.

Met de asynchrone versies kunt u processen opstarten die op de achtergrond worden uitgevoerd en gebeurtenissen verzenden wanneer ze zijn voltooid (of wanneer er een fout optreedt). Andere code kan worden uitgevoerd terwijl deze asynchrone achtergrondprocessen plaatsvinden. Met asynchrone versies van de bewerkingen moet u gebeurtenislistenerfuncties instellen met behulp van de methode `addEventListener()` van het File- of FileStream-object dat de functie oproept.

Met asynchrone versies kunt u eenvoudiger code schrijven die niet afhankelijk is van het instellen van gebeurtenislisteners. Aangezien geen andere code kan worden uitgevoerd terwijl een synchrone methode wordt uitgevoerd, is het mogelijk dat belangrijke processen worden onderbroken, zoals de rendering van weergaveobjecten en animatie.

Werken met File-objecten in AIR

Adobe AIR 1.0 of hoger

Een File-object is een aanwijzer naar een bestand of map in het bestandssysteem.

De klasse File is een uitbreiding van de klasse FileReference. De klasse FileReference, die beschikbaar is in Adobe® Flash® Player en AIR, vertegenwoordigt een aanwijzer voor een bestand. De klasse File voegt eigenschappen en methoden toe die uit veiligheidsoverwegingen niet worden blootgesteld in Flash Player (in een SWF-bestand dat wordt uitgevoerd in een browser).

Informatie over de klasse File

Adobe AIR 1.0 of hoger

U kunt de klasse File voor het volgende gebruiken:

- het ophalen van het pad naar speciale mappen, inclusief de gebruikersmap, de documentenmap van de gebruiker, de map van waaruit de toepassing is gestart en de toepassingsmap;
- het kopiëren van bestanden en mappen;
- het verplaatsen van bestanden en mappen;
- het verwijderen van bestanden en mappen (of het verplaatsen ervan naar de prullenbak);
- het weergeven van lijsten van bestanden en mappen die zich bevinden in een map;
- het maken van tijdelijke bestanden en mappen.

Als een File-object eenmaal een bestandsmap aanwijst, kunt u dat object gebruiken om bestandsgegevens te lezen en te schrijven met behulp van de klasse FileStream.

Een File-object kan het pad aanwijzen van een bestand dat of een map die nog niet bestaat. U kunt zo'n File-object gebruiken bij het maken van een bestand of map.

Paden van File-objecten

Adobe AIR 1.0 of hoger

Ieder File-object heeft twee eigenschappen die het pad ervan definiëren:

Eigenschap	Beschrijving
`nativePath`	Geeft het platformspecifieke pad naar een bestand op. In Windows kan een pad bijvoorbeeld "C:\Voorbeeldmap\test.txt" zijn, terwijl in Mac OS het pad "/Voorbeeldmap/test.txt" is. De eigenschap `nativePath` gebruikt de backslash (\) als scheidingsteken voor mappen in Windows, en de schuine streep (/) in Mac OS en Linux.
`url`	Het URL-schema file kan dit gebruiken om een bestand aan te wijzen. In Windows kan een pad bijvoorbeeld "file:///C:/Voorbeeldmap/test.txt" zijn, terwijl in Mac OS het pad "file:///Voorbeeldmap/test.txt" is. De runtime bevat nog andere speciale URL-schema's behalve `file`; deze worden beschreven in "Ondersteunde URL-protocollen voor AIR" op pagina 715

De File-klasse bevat statische eigenschappen voor het aanwijzen van standaardmappen op Mac OS, Windows en Linux. Deze eigenschappen bevatten:

- `File.applicationStorageDirectory`—een opslagmap die uniek is voor elke geïnstalleerde AIR-toepassing.. Deze map is geschikt voor het opslaan van dynamische toepassingselementen en gebruikersvoorkeuren. U kunt grote hoeveelheden gegevens beter ergens anders opslaan.

  Op Android en iOS wordt de toepassingsopslagmap verwijderd wanneer de toepassing wordt verwijderd of wanneer de gebruiker besluit toepassingsgegevens te wissen. Op andere platformen is dit niet het geval.

- `File.applicationDirectory`—de map waarin de toepassing is geïnstalleerd (samen met geïnstalleerde onderdelen). Op sommige besturingssystemen wordt de toepassing opgeslagen in één pakketbestand in plaats van in een fysieke map. In dat geval is de inhoud wellicht niet bereikbaar via het geïntegreerde pad. De toepassingsmap heeft het kenmerk Alleen-lezen.
- `File.desktopDirectory`—de bureaubladmap van de gebruiker. Als een platform geen bureaubladmap definieert, wordt een andere locatie in het bestandssysteem gebruikt.
- `File.documentsDirectory`—de documentmap van de gebruiker. Als een platform geen documentmap definieert, wordt een andere locatie in het bestandssysteem gebruikt.
- `File.userDirectory`—de gebruikersmap. Als een platform geen gebruikersmap definieert, wordt een andere locatie in het bestandssysteem gebruikt.

Opmerking: *Wanneer een platform geen standaardlocatie definieert voor bureaublad-, document- of gebruikersmappen, kunnen* `File.documentsDirectory`*,* `File.desktopDirectory`*, en* `File.userDirectory` *naar dezelfde map verwijzen.*

Deze eigenschappen hebben verschillende waarden in verschillende besturingssystemen. Mac en Windows hebben bijvoorbeeld elk een ander geïntegreerd pad naar de bureaubladmap van de gebruiker. De eigenschap `File.desktopDirectory` verwijst echter naar een geschikt mappad op elk platform. Om toepassingen te schrijven die goed op alle platformen werken, gebruikt u deze eigenschappen als de basis voor verwijzen naar andere mappen en bestanden die door de toepassing worden gebruikt. Gebruik de `resolvePath()`-methode om het pad te verfijnen. Deze code wijst bijvoorbeeld naar het bestand preferences.xml in de opslagmap van de toepassing:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

Hoewel u met de File-klasse naar een bepaald bestandspad kunt wijzen, kan dit leiden naar toepassingen die niet op alle platformen werken. Het pad C:\Documents and Settings\joe\ werkt bijvoorbeeld alleen op Windows. Om die reden is het beter om de statische eigenschappen van de File-klasse te gebruiken, zoals `File.documentsDirectory`.

Gebruikelijke maplocaties

Platform	Type map	Standaardlocatie voor bestandssysteem
Android	Toepassing	`/data/data/`
	Opslag van toepassingen	`/data/data/air.applicationID/filename/Local Store`
	Cache	/data/data/applicationID/cache
	Desktop	`/mnt/sdcard`
	Documenten	`/mnt/sdcard`
	Tijdelijke opslag	`/data/data/applicationID/cache/FlashTmp.randomString`
	Gebruiker	`/mnt/sdcard`
iOS	Toepassing	`/var/mobile/Applications/uid/filename.app`
	Opslag van toepassingen	`/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store`
	Cache	/var/mobile/Applications/uid/Library/Caches
	Desktop	niet toegankelijk
	Documenten	`/var/mobile/Applications/uid/Documents`
	Tijdelijke opslag	`/private/var/mobile/Applications/uid/tmp/FlashTmpNNN`
	Gebruiker	niet toegankelijk
Linux	Toepassing	`/opt/bestandsnaam/share`
	Opslag van toepassingen	`/home/userName/.appdata/applicationID/Local Store`
	Desktop	`/home/userName/Desktop`
	Documenten	`/home/userName/Documents`
	Tijdelijke opslag	`/tmp/FlashTmp.randomString`
	Gebruiker	`/home/userName`
Mac	Toepassing	`/Applications/filename.app/Contents/Resources`
	Opslag van toepassingen	`/Users/`***`gebruikersnaam`***`/Library/Preferences/`***`toepassings-id`***`/Local Store` (AIR 3.2 en lager) ***`pad`***`/Library/Application Support/`***`toepassings-id`***`/Local Store` (AIR 3.3 en hoger), waarbij het pad ofwel `/Users/`***`gebruikersnaam`***`/Library/Containers/`***`bundle-id`***`/Data` is (in een sandbox-omgeving), of `/Users/gebruikersnaam` (buiten een sandbox-omgeving)
	Cache	/Users/userName/Library/Caches
	Desktop	`/Users/`***`gebruikersnaam`***`/Desktop`
	Documenten	`/Users/`***`gebruikersnaam`***`/Documents`
	Tijdelijke opslag	`/private/var/folders/JY/randomString/TemporaryItems/FlashTmp`
	Gebruiker	`/Users/`***`gebruikersnaam`***

Platform	Type map	Standaardlocatie voor bestandssysteem
Windows	Toepassing	`C:\Program Files\filename`
	Opslag van toepassingen	`C:\Documents and settings\userName\ApplicationData\applicationID\Local Store`
	Cache	C:\Documents and settings\userName\Local Settings\Temp
	Desktop	`C:\Documents and settings\userName\Desktop`
	Documenten	`C:\Documents and settings\userName\My Documents`
	Tijdelijke opslag	`C:\Documents and settings\userName\Local Settings\Temp\randomString.tmp`
	Gebruiker	`C:\Documents and settings\userName`

De feitelijke native paden voor deze mappen variëren per besturingssysteem en computerconfiguratie. In deze tabel ziet u typische voorbeelden van paden. U dient altijd met de desbetreffende statische eigenschappen voor File-klasse naar deze mappen te verwijzen, zodat uw toepassing op elk platform naar behoren functioneert. In een feitelijke AIR-toepassing worden de waarden voor `applicationID` en `filename` die in de tabel staan, ontleend aan de toepassingsbeschrijving. Als u een uitgever-ID opgeeft in de toepassingsbeschrijving, wordt deze aan de toepassings-ID toegevoegd in deze paden. De waarde voor `userName` is de accountnaam van de gebruiker die de installatie uitvoert.

File-objecten een map laten aanwijzen

Adobe AIR 1.0 of hoger

Er zijn verschillende manieren om een File-object zo in te stellen dat het een map aanwijst.

De homemap van de gebruiker aanwijzen

Adobe AIR 1.0 of hoger

U kunt een File-object verwijzen naar de thuismap van de gebruiker. Met de volgende code stelt u een File-object zo in dat het de submap AIR Test van de homemap aanwijst:

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

De documentenmap van de gebruiker aanwijzen

Adobe AIR 1.0 of hoger

U kunt een File-object de documentenmap van de gebruiker laten aanwijzen. Met de volgende code stelt u een File-object zo in dat het de submap AIR Test van de documentenmap aanwijst:

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

De bureaubladmap aanwijzen

Adobe AIR 1.0 of hoger

U kunt een File-object het bureaublad laten aanwijzen. Met de volgende code stelt u een File-object zo in dat het de submap AIR Test van het bureaublad aanwijst:

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

De opslagmap van een toepassing aanwijzen

Adobe AIR 1.0 of hoger

U kunt een File-object de opslagmap van een toepassing laten aanwijzen. Bij iedere AIR-toepassing is er een uniek gekoppeld pad dat de opslagmap van de toepassing definieert. Deze map is uniek voor iedere toepassing en iedere gebruiker. U kunt deze map gebruiken om gebruikers- en toepassingsspecifieke gegevens op te slaan (zoals gebruikersgegevens of voorkeursbestanden). Met de volgende code wijst een File-object bijvoorbeeld een voorkeurenbestand met de naam prefs.xml aan dat zich bevindt in de opslagmap van de toepassing:

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

De locatie van de opslagmap van de toepassing is meestal gebaseerd op de gebruikersnaam en de toepassings-id. U vindt hier de volgende bestandssysteemlocaties om u te helpen met de foutopsporing in uw toepassing. Gebruik altijd de eigenschap `File.applicationStorage` of het URI-schema `app-storage:` om bestanden op te lossen in deze map:

- Op Mac OS: varieert per AIR-versie:

  AIR 3.2 en eerder: `/Users/gebruikersnaam/Library/Preferences/`*`toepassings-id`*`/Local Store/`

  AIR 3.3 en hoger: *`pad`*`/Library/Application Support/`*`applicationID`*`/Local Store`, waarbij *`pad`* ofwel `/Users/`*`gebruikersnaam`*`/Library/Containers/`*`bundle-id`*`/Data` is (in een sandbox-omgeving), of `/Users/`*`gebruikersnaam`* (buiten een sandbox-omgeving)

  Bijvoorbeeld (AIR 3.2):

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- In Windows, in de map Documents and Settings in:

  C:\Documents and Settings\gebruikersnaam`\Application Data\`*`applicationID`*`\Lokale opslag\`

  Bijvoorbeeld:

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- Op Linux—In:

  `/home/`*`gebruikersnaam`*`/.appdata/`*`toepassings-id`*`//Local Store/`

  Bijvoorbeeld:

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- Android—In:

  /data/data/`androidPackageID/applicationID/Lokale opslag`

  Bijvoorbeeld:

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

Opmerking: *Als een toepassing een uitgevers-id heeft, wordt deze id ook gebruikt als onderdeel van het pad naar de opslagmap van de toepassing.*

De URL (en de eigenschap `url`) van een File-object dat is gemaakt met `File.applicationStorageDirectory`, gebruikt het URL-schema `app-storage` (zie "Ondersteunde URL-protocollen voor AIR" op pagina 715). Zie het volgende voorbeeld:

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

De toepassingsmap aanwijzen

Adobe AIR 1.0 of hoger

U kunt een File-object de map laten aanwijzen waarin de toepassing is geïnstalleerd. Deze wordt de toepassingsmap genoemd. U kunt verwijzen naar deze map met behulp van de eigenschap `File.applicationDirectory`. U kunt deze map gebruiken om het toepassingsdescriptorbestand of andere resources die met de toepassing zijn geïnstalleerd, te onderzoeken. Met de volgende code wijst een File-object bijvoorbeeld de map *images* in de toepassingsmap aan:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

De URL (en de eigenschap `url`) van een File-object dat is gemaakt met `File.applicationDirectory`, gebruikt het URL-schema `app` (zie "Ondersteunde URL-protocollen voor AIR" op pagina 715). Zie het volgende voorbeeld:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

Opmerking: *Op Android zijn de bestanden in het toepassingspakket niet toegankelijk via het `nativePath`. De eigenschap `nativePath` is een lege tekenreeks. Gebruik altijd de URL in plaats van een geïntegreerd pad voor toegang tot de bestanden in de toepassingsmap.*

De cachemap aanwijzen

Adobe AIR 3.6 of hoger

U kunt een File-object laten wijzen naar de tijdelijke map of cachemap van het besturingssysteem met de `File.cacheDirectory`-eigenschap. Deze map bevat tijdelijke bestanden die niet vereist zijn om de toepassing te laten uitvoeren en veroorzaken geen problemen of gegevensverlies voor de gebruiker als ze worden verwijderd.

In de meeste besturingssystemen is de cachemap een tijdelijke map. Op iOS komt de cachemap overeen met de map Caches van de toepassingsbibliotheek. Er worden geen onlineback-ups gemaakt van de bestanden in deze map. Ze kunnen dus mogelijk worden verwijderd door het besturingssysteem als het apparaat over onvoldoende opslagruimte beschikt. Zie "Back-ups en caching van bestanden beheren" op pagina 715 voor meer informatie.

Verwijzen naar het hoofdbestandssysteem

Adobe AIR 1.0 of hoger

De methode `File.getRootDirectories()` geeft een lijst weer van alle hoofdvolumes (zoals C:) en alle gekoppelde volumes op een Windows-computer. Op Mac OS en Linux retourneert deze methode altijd de unieke hoofdmap van de computer (de map "/"). De methode `StorageVolumeInfo.getStorageVolumes()` verschaft meer informatie over gekoppelde opslagvolumes (zie "Werken met opslagvolumes" op pagina 727).

Opmerking: *De hoofdmap van het bestandssysteem kan niet worden gelezen op Android. Voor een File-object dat naar de map met het geïntegreerde pad verwijst, wordt "/" geretourneerd, maar de eigenschappen van dat object hebben niet de juiste waarden. `spaceAvailable` is bijvoorbeeld altijd 0.*

Een expliciete map aanwijzen

Adobe AIR 1.0 of hoger

U kunt het File-object een expliciete map laten aanwijzen door de eigenschap `nativePath` van het File-object in te stellen, zoals in het volgende voorbeeld (in Windows):

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

Belangrijk: Op deze manier verwijzen naar een expliciet pad kan leiden naar code die niet op verschillende platformen werkt. Het vorige voorbeeld werkt bijvoorbeeld alleen op Windows. U kunt de statische eigenschappen van het File-object gebruiken, zoals `File.applicationStorageDirectory`, om een map te zoeken die op verschillende platformen werkt. Gebruik de `resolvePath()`-methode (zie de volgende sectie) om naar een relatief pad te navigeren.

Navigeren naar relatieve paden

Adobe AIR 1.0 of hoger

U kunt de methode `resolvePath()` gebruiken om een pad te verkrijgen dat is gebaseerd op een ander gegeven pad. Met de volgende code stelt u bijvoorbeeld een File-object zo in dat het de submap AIR Test van de homemap van de gebruiker aanwijst:

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

U kunt ook de eigenschap `url` van een File-object gebruiken om dit een map te laten aanwijzen op basis van een URL-tekenreeks, zoals bij het volgende:

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

Zie "Bestandspaden wijzigen" op pagina 714 voor meer informatie.

De gebruiker naar de gewenste map laten bladeren en deze selecteren

Adobe AIR 1.0 of hoger

De klasse File bevat de methode `browseForDirectory()`, die een systeemdialoogvenster presenteert waarin de gebruiker de map kan selecteren die aan het object moet worden toegewezen. De methode `browseForDirectory()` is asynchroon. Het File-object verzendt de gebeurtenis `select` als de gebruiker een map selecteert en op de knop Openen klikt. De gebeurtenis `cancel` wordt verzonden als de gebruiker op de knop Annuleren klikt.

Met de volgende code kan de gebruiker bijvoorbeeld een map selecteren en wordt na selectie het pad van de map als uitvoergegeven ingesteld:

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

Opmerking: *De methode `browseForDirectory()` wordt niet ondersteund op Android. Het aanroepen van deze methode heeft geen effect, er wordt meteen een annuleringsgebeurtenis verzonden. Als u gebruikers wilt toestaan een map te selecteren, dient u een aangepast, door de toepassing gedefinieerd dialoogvenster te gebruiken.*

De map aanwijzen van waaruit de toepassing is opgeroepen

Adobe AIR 1.0 of hoger

U kunt de locatie opvragen van de map van waaruit een toepassing is opgeroepen door de eigenschap `currentDirectory` van het InvokeEvent-object te controleren dat wordt verzonden wanneer de toepassing wordt opgeroepen. Zie "Opdrachtregelargumenten vastleggen" op pagina 929 voor meer informatie.

File-objecten een bestand laten aanwijzen

Adobe AIR 1.0 of hoger

Er zijn verschillende manieren om het bestand in te stellen dat door een File-object wordt aangewezen.

Expliciete bestandspaden aanwijzen

Adobe AIR 1.0 of hoger

Belangrijk: verwijzen naar een expliciet pad kan leiden tot code die niet op verschillende platformen werkt. Het pad C:/foo.txt werkt bijvoorbeeld alleen op Windows. U kunt de statische eigenschappen van het File-object gebruiken, zoals `File.applicationStorageDirectory`, om een map te zoeken die op verschillende platformen werkt. Gebruik de `resolvePath()`-methode (zie "Bestandspaden wijzigen" op pagina 714) om naar een relatief pad te navigeren.

U kunt de eigenschap `url` van een File-object gebruiken om dit een bestand of map te laten aanwijzen op basis van een URL-tekenreeks, zoals in het volgende voorbeeld:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

U kunt de URL ook doorgeven naar de constructorfunctie `File()`, zoals in het volgende voorbeeld:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

De eigenschap `url` retourneert altijd de URI-gecodeerde versie van de URL (spaties worden bijvoorbeeld vervangen door `"%20"`):

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

U kunt ook de eigenschap `nativePath` van een File-object gebruiken om een expliciet pad in te stellen. Als de volgende code bijvoorbeeld wordt uitgevoerd op een Windows-computer, wordt een File-object ingesteld op het bestand test.txt in de submap AIR Test van het C:-station:

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

U kunt dit pad ook doorgeven naar de constructorfunctie `File()`, zoals in het volgende voorbeeld:

```
var file:File = new File("C:/AIR Test/test.txt");
```

Gebruik het slashteken (/) als het padscheidingsteken voor de `nativePath`-eigenschap. Op Windows kunt u ook het backslashteken (\) gebruiken, maar dit leidt tot toepassingen die niet op verschillende platformen werken.

Zie "Bestandspaden wijzigen" op pagina 714 voor meer informatie.

Opsommingen maken van bestanden in een map

Adobe AIR 1.0 of hoger

U kunt de methode `getDirectoryListing()` van een File-object gebruiken om een array op te halen van File-objecten die bestanden en submappen aanwijzen op het hoofdniveau van een map. Zie "Opsommingen maken van mappen" op pagina 722 voor meer informatie.

De gebruiker naar het gewenste bestand laten bladeren en dit selecteren

Adobe AIR 1.0 of hoger

De klasse File bevat de volgende methoden die een systeemdialoogvenster presenteren waarin de gebruiker een bestand kan selecteren dat aan het object moet worden toegewezen:

- `browseForOpen()`
- `browseForSave()`
- `browseForOpenMultiple()`

Deze methoden zijn allemaal asynchroon. De methoden `browseForOpen()` en `browseForSave()` verzenden de gebeurtenis select wanneer de gebruiker een bestand selecteert (of een doelpad, bij het gebruik van browseForSave()). Met de methoden `browseForOpen()` en `browseForSave()` wijst het File-doelobject na selectie de geselecteerde bestanden aan. De methode `browseForOpenMultiple()` verzendt de gebeurtenis `selectMultiple` wanneer de gebruiker bestanden selecteert. De gebeurtenis `selectMultiple` is van het type FileListEvent. Dit type heeft de eigenschap `files`, die bestaat uit een array van File-objecten (die de geselecteerde bestanden aanwijzen).

De volgende code presenteert de gebruiker bijvoorbeeld met het dialoogvenster "Openen" waarin deze een bestand kan selecteren:

```
 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);
}
```

Als er in de toepassing een ander dialoogvenster voor het bladeren naar mappen of bestanden open is wanneer u een bladermethode oproept, produceert de runtime een uitzonderingsfout.

Opmerking: *Op Android kunt u alleen afbeeldings-, video- en audiobestanden selecteren met de methoden `browseForOpen()` en `browseForOpenMultiple()`. Ook in het dialoogvenster browseForSave() worden alleen mediabestanden weergegeven, ook al kan de gebruiker een willekeurige bestandsnaam invoeren. Als u andere bestandstypen wilt openen, kunt u beter aangepaste dialoogvensters gebruiken in plaats van deze methoden.*

Bestandspaden wijzigen

Adobe AIR 1.0 of hoger

U kunt ook het pad van een bestaand File-object wijzigen door de methode `resolvePath()` op te roepen, of door de eigenschap `nativePath` of `url` van het object op te roepen, zoals in de volgende voorbeelden (voor 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
```

Wanneer u de eigenschap `nativePath` gebruikt, moet u het slashteken (/) gebruiken als mapscheidingsteken. Op Windows kunt u ook het backslashteken (\) gebruiken, maar we raden u dit ten sterkste af, omdat dit leidt tot code die niet op verschillende platformen werkt.

Ondersteunde URL-protocollen voor AIR

Adobe AIR 1.0 of hoger

In AIR kunt u een van de volgende URL-protocollen gebruiken om de `url`-eigenschap van een File-object te definiëren:

URL-schema	Beschrijving
file	Wordt gebruikt om een pad op te geven dat is gebaseerd op de hoofdmap van het bestandssysteem. Bijvoorbeeld: `file:///c:/AIR Test/test.txt` De URL-standaard bepaalt dat een bestands-URL de volgende vorm heeft: `file://<host>/<pad>`. Als speciaal geval kan `<host>` een lege tekenreeks zijn. Dit wordt geïnterpreteerd als "de computer van waaruit de URL wordt geïnterpreteerd". Om deze reden hebben bestands-URL's vaak drie schuine strepen (///).
app	Wordt gebruikt om een pad op te geven dat is gebaseerd op de hoofdmap (de map met het bestand application.xml van de geïnstalleerde toepassing) van de geïnstalleerde toepassing. Het volgende pad wijst bijvoorbeeld de submap images van de map van de geïnstalleerde toepassing aan: `app:/images`
app-storage	Wordt gebruikt om een pad op te geven dat is gebaseerd op de opslagmap van de toepassing. Voor elke geïnstalleerde toepassing definieert AIR een unieke opslagmap. Dit is de plaats waar gegevens worden opgeslagen die eigen zijn aan die toepassing. Het volgende pad verwijst bijvoorbeeld naar het bestand prefs.xml in de submap settings van de opslagmap van een toepassing: `app-storage:/settings/prefs.xml`

Back-ups en caching van bestanden beheren

Adobe AIR 3.6 of hoger, alleen iOS en OS X

Bepaalde besturingssystemen, met name iOS en Mac OS X, bieden gebruikers de mogelijkheid om automatisch back-ups te maken van toepassingsbestanden naar een externe opslagruimte. Bovendien gelden er op iOS beperkingen over het al dan niet maken van back-ups van bestanden en waar bestanden met verschillende doeleinden kunnen worden opgeslagen.

Het volgende vat samen hoe de richtlijnen van Apple voor het maken van back-ups en het opslaan van bestanden moeten worden nageleefd. Zie de volgende secties voor meer informatie.

- Om op te geven dat er geen back-up van een bestand moet worden gemaakt en dat (alleen op iOS) een bestand kan worden verwijderd door het besturingssysteem als het apparaat over onvoldoende opslagruimte beschikt, slaat u het bestand op in de cachemap (`File.cacheDirectory`). Dit is de voorkeursopslaglocatie op iOS en deze moet worden gebruikt voor de meeste bestanden die opnieuw kunnen worden gegenereerd of gedownload.
- Om op te geven dat er geen back-up van een bestand moet worden gemaakt, maar dat het bestand niet moet worden verwijderd door het besturingssysteem, slaat u het bestand op in een van de mappen van de toepassingsbibliotheek, zoals de toepassingsopslagmap (`File.applicationStorageDirectory`) of de documentenmap (`File.documentsDirectory`). Stel de `preventBackup`-eigenschap van het File-object in op `true`. Dit is verplicht door Apple voor inhoud die opnieuw kan worden gegenereerd of gedownload, maar die vereist is voor een goede werking van uw toepassing tijdens offlinegebruik.

Bestanden opgeven voor back-up

Om back-upruimte te besparen en het gebruik van de netwerkbandbreedte te verminderen, geven de richtlijnen van Apple voor iOS- en Mac-toepassingen op dat alleen bestanden met door gebruikers ingevoerde gegevens of gegevens die niet op een andere manier opnieuw kunnen worden gegenereerd of gedownload, moeten worden opgegeven voor back-up.

Standaard wordt van alle bestanden in de toepassingsbibliotheekmappen een back-up gemaakt. Op Mac OS X is dit de opslagmap van de toepassing. Op iOS omvat dit de opslagmap van de toepassing, de toepassingsmap, de bureaubladmap, de documentenmap en de gebruikersmap (omdat deze mappen zijn toegewezen aan toepassingsbibliotheekmappen op iOS). Bijgevolg worden standaard back-ups gemaakt naar de serveropslag van alle bestanden in deze mappen.

Als u een bestand opslaat op een van deze locaties die door uw toepassing opnieuw kunnen worden gemaakt, moet u het bestand markeren zodat het besturingssysteem weet dat het hiervan geen back-up moet maken. Stel de `preventBackup`-eigenschap van het File-object in op `true` om aan te duiden dat er geen back-up moet worden gemaakt van een bestand.

Merk op dat op iOS een bestand in een van de toepassingsbibliotheekmappen wordt gemarkeerd als een permanent bestand dat niet mag worden verwijderd door het besturingssysteem, zelfs als de `preventBackup`-eigenschap van dat bestand is ingesteld op `true`.

Caching en verwijderen van bestanden beheren

De richtlijnen van Apple voor iOS-bestanden geven aan dat inhoud die opnieuw kan worden gegenereerd, zoveel mogelijk ter beschikking moet worden gesteld van het besturingssysteem om te worden verwijderd voor het geval dat het apparaat over onvoldoende opslagruimte beschikt.

Op iOS worden bestanden in de toepassingsbibliotheekmappen (zoals de opslagmap van de toepassing of de documentenmap) als permanent gemarkeerd en worden deze niet verwijderd door het besturingssysteem.

Sla bestanden op die door de toepassing opnieuw kunnen worden gegenereerd en zonder problemen kunnen worden verwijderen voor het geval dat de cachemap van de toepassing over onvoldoende opslagruimte beschikt. U krijgt toegang tot de cachemap met de statische eigenschap `File.cacheDirectory`.

Op iOS stemt de cachemap overeen met de cachemap van de toepassing (<Application Home>/Library/Caches). Op andere besturingssystemen wordt deze map toegewezen aan een vergelijkbare map. Op Mac OS X, bijvoorbeeld, wordt deze ook toegewezen aan de map Caches in de toepassingsbibliotheek. Op Android wordt de cachemap toegewezen aan de cachemap van de toepassing. Op Windows wordt de cachemap toegewezen aan de map temp van het besturingssysteem. Op zowel Android als Windows is dit dezelfde map die wordt geopend door de methoden `createTempDirectory()` en `createTempFile()` van de File-klasse aan te roepen.

Het relatieve pad tussen twee bestanden opzoeken

Adobe AIR 1.0 of hoger

U kunt de methode `getRelativePath()` gebruiken om het relatieve pad tussen twee bestanden op te zoeken:

```
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
```

De tweede parameter van de methode `getRelativePath()`, de parameter `useDotDot` maakt het mogelijk dat de syntaxis `..` wordt geretourneerd in resultaten om bovenliggende mappen aan te geven:

```
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
```

Canonieke versies van bestandsnamen verkrijgen

Adobe AIR 1.0 of hoger

Bestandsnamen en padnamen zijn niet hoofdlettergevoelig in Windows en Mac OS. In het volgende voorbeeld wijzen twee File-objecten hetzelfde bestand aan:

```
File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

Documenten en mapnamen maken echter wel gebruik van hoofdletters en kleine letters. Bij het volgende wordt ervan uitgegaan dat de documentenmap een map met de naam AIR Test bevat, zoals in de volgende voorbeelden:

```
var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

De methode `canonicalize` converteert het `nativePath`-object zodat de juiste hoofdletters en kleine letters worden gebruikt voor de bestands- of mapnaam. Als er bij hoofdlettergevoelige systemen (zoals Linux) meerdere bestanden zijn waarvan de namen alleen verschillen wat betreft hoofdlettergebruik, past de methode `canonicalize()` het pad aan zodat het past bij het eerste bestand dat wordt aangetroffen (in de volgorde die wordt bepaald door het bestandssysteem).

U kunt de methode `canonicalize()` ook gebruiken om korte bestandsnamen ("8.3"-namen) te converteren naar lange bestandsnamen in Windows, zoals in de volgende voorbeelden:

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

Werken met pakketten en symbolische koppelingen

Adobe AIR 1.0 of hoger

Verschillende besturingssystemen ondersteunen pakketbestanden en symbolische koppelingsbestanden:

Pakketten In Mac OS kunnen mappen worden aangewezen als pakketten. Ze worden dan in de Finder van Mac OS als één bestand en niet als een map weergegeven.

Symbolische koppelingen Mac OS, Linux en Windows Vista ondersteunen symbolische koppelingen. Met symbolische koppelingen kan een bestand naar een ander bestand of een map op de schijf verwijzen. Symbolische koppelingen zijn vergelijkbaar met aliassen, maar ze zijn niet hetzelfde. Een alias wordt altijd als een bestand (en niet als een map) gerapporteerd, en het lezen van of schrijven naar een alias of snelkoppeling is nooit van invloed op het oorspronkelijke bestand of de map die deze aanwijst. Een symbolische koppeling daarentegen gedraagt zich precies hetzelfde als het bestand of de map die deze koppeling aanwijst. Een symbolische koppeling kan als een bestand of een map worden gerapporteerd. Het lezen van of schrijven naar een symbolische koppeling is van invloed op het bestand of de map die deze koppeling aanwijst, en niet op de symbolische koppeling zelf. Verder is onder Windows de eigenschap `isSymbolicLink` voor een object File dat verwijst naar een junction point (wordt gebruikt in het NTFS-bestandssysteem) ingesteld op `true` .

De klasse File bevat de eigenschappen `isPackage` en `isSymbolicLink`, waarmee kan worden gecontroleerd of een File-object naar een pakket dan wel naar een symbolische koppeling verwijst.

De volgende code doorzoekt de bureaubladmap van de gebruiker en geeft een lijst weer van de submappen die *geen* pakketten zijn:

```
 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);
    }
}
```

De volgende code doorzoekt de bureaubladmap van de gebruiker en geeft een lijst weer van de bestanden en mappen die *geen* symbolische koppelingen zijn:

```
 var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

De methode `canonicalize()` wijzigt het pad van een symbolische koppeling zodat deze het bestand of de map aanwijst waarnaar de koppeling verwijst. De volgende code doorzoekt de bureaubladmap van de gebruiker en rapporteert de paden waarnaar wordt verwezen door bestanden die symbolische koppelingen zijn:

```
 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);
    }
}
```

De beschikbare ruimte op een volume bepalen

Adobe AIR 1.0 of hoger

De eigenschap `spaceAvailable` van een File-object is de beschikbare ruimte op de File-locatie, in bytes. De volgende code controleert bijvoorbeeld de ruimte die beschikbaar is in de opslagmap van de toepassing:

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

Als het File-object verwijst naar een map, geeft de eigenschap `spaceAvailable` de ruimte aan die in die map beschikbaar is voor bestanden. Als het File-object verwijst naar een bestand, geeft de eigenschap `spaceAvailable` de ruimte aan die beschikbaar is als het bestand in grootte toeneemt. Als de bestandslocatie niet bestaat, wordt de eigenschap `spaceAvailable` ingesteld op 0. Als het File-object verwijst naar een symbolische koppeling, wordt de eigenschap `spaceAvailable` ingesteld op de ruimte die beschikbaar is op de locatie die de symbolische koppeling aanwijst.

De ruimte die beschikbaar is voor een bestand of map, is in het algemeen hetzelfde als de ruimte die beschikbaar is op het volume dat het bestand of de map bevat. Bij de beschikbare ruimte kan echter rekening worden gehouden met accountquota's en groottebeperkingen per map.

Het toevoegen van een bestand of map aan een volume vereist in het algemeen meer ruimte dan de werkelijke grootte van het bestand of de grootte van de inhoud van de map. Het besturingssysteem kan bijvoorbeeld meer ruimte nodig hebben om indexgegevens op te slaan. Het is ook mogelijk dat de vereiste schijfsectoren meer ruimte gebruiken. Bovendien verandert de beschikbare ruimte dynamisch. U kunt dus niet verwachten dat de volledige gerapporteerde schijfruimte kan worden toegewezen voor bestandsopslag. Zie “Lezen van en schrijven naar bestanden” op pagina 728 voor meer informatie.

De methode `StorageVolumeInfo.getStorageVolumes()` verschaft meer informatie over gekoppelde opslagvolumes (zie “Werken met opslagvolumes” op pagina 727).

Bestanden openen met de standaardsysteemtoepassing

Adobe AIR 2 of hoger

In AIR 2 kunt u een bestand openen met de toepassing die hiervoor bij het besturingssysteem is geregistreerd. Een AIR-toepassing kan bijvoorbeeld een DOC-bestand openen met de toepassing die hiervoor geregistreerd is. Gebruik de methode `openWithDefaultApplication()` van een File-object om het bestand te openen. Met de volgende code wordt bijvoorbeeld het bestand test.doc geopend op het bureaublad van de gebruiker. Dit bestand wordt geopend met de standaardtoepassing voor DOC-bestanden:

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

Opmerking: *Op het Linux-platform wordt de standaardtoepassing van een bestand bepaald door het MIME-type van het bestand en niet door de extensie van de bestandsnaam.*

Met de volgende code kan de gebruiker naar een MP3-bestand navigeren en dit openen in de standaardtoepassing voor het afspelen van MP3-bestanden:

```
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();
}
```

De methode `openWithDefaultApplication()` kan niet worden gebruikt voor bestanden in de toepassingsmap.

Er zijn een aantal bestanden die in Air niet kunnen worden geopend met de methode `openWithDefaultApplication()`. In Windows kunt u bestanden met bepaalde bestandstypen, zoals bestanden met de indeling .exe of .bat, niet openen. Bij Mac OS- en Linux-systemen voorkomt AIR dat u bestanden voor een bepaalde toepassing kunt openen. (Hieronder vallen Terminal en AppletLauncher op Mac OS-systemen en csh, bash of ruby op Linux-systemen.) Er wordt een uitzondering gegenereerd als u een van deze bestanden met de methode `openWithDefaultApplication()` probeert te openen. Zie in de taalnaslag de vermelding voor de methode `File.openWithDefaultApplication()` voor een complete lijst met bestandstypen die niet kunnen worden geopend.

Opmerking: *Deze beperking geldt niet voor AIR-toepassingen die met een native installatieprogramma zijn geïnstalleerd (een uitgebreide desktoptoepassing).*

Informatie over het bestandssysteem ophalen

Adobe AIR 1.0 of hoger

De klasse File bevat de volgende statische eigenschappen die nuttige informatie over het bestandssysteem kunnen geven:

Eigenschap	Beschrijving
`File.lineEnding`	Het regeleindeteken dat wordt gebruikt door het hostbesturingssysteem. In Mac OS en Linux is dit het line feed-teken. In Windows is dit het carriage return-teken, gevolgd door het line feed-teken.
`File.separator`	Het teken dat in het hostbesturingssysteem wordt gebruikt om de onderdelen van paden te scheiden. In Mac OS en Linux is dit de schuine streep (/). In Windows is dit de backslash (\).
`File.systemCharset`	De standaardcodering die door het hostbesturingssysteem wordt gebruikt voor bestanden. Dit heeft betrekking op de tekenset die door het besturingssysteem wordt gebruikt en die correspondeert met de taal.

De `Capabilities`-klasse bevat ook nuttige systeeminformatie die nuttig kan zijn wanneer u met bestanden werkt:

Eigenschap	Beschrijving
Capabilities.hasIME	Geeft aan of de speler op een systeem wordt uitgevoerd waarop een invoermethode-editor (IME) is geïnstalleerd (`true`) of niet (`false`).
Capabilities.language	Geeft de taalcode op van het systeem waarop de speler wordt uitgevoerd.
Capabilities.os	Geeft het huidige besturingssysteem op.

Opmerking: *Wees voorzichtig wanneer u `Capabilities.os` gebruikt om systeemkenmerken te bepalen. Als er een specifiekere eigenschap bestaat om een systeemkenmerk te bepalen, raden we u aan om deze te gebruiken. Anders loopt u het risico dat u code schrijft die niet correct werkt op alle platformen. Neem bijvoorbeeld de volgende code:*

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

Deze code leidt tot problemen op Linux. U kunt beter de `File.separator`-eigenschap gebruiken.

Werken met mappen

Adobe AIR 1.0 of hoger

De runtime biedt u mogelijkheden om te werken met mappen in het lokale bestandssysteem.

Zie "File-objecten een map laten aanwijzen" op pagina 709 voor meer informatie over het maken van File-objecten die mappen aanwijzen.

Mappen maken

Adobe AIR 1.0 of hoger

Met de methode `File.createDirectory()` kunt u een map maken. Met de volgende code maakt u bijvoorbeeld een map met de naam AIR Test die een submap is van de homemap van de gebruiker:

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

Als die map al bestaat, doet de methode `createDirectory()` verder niets.

In bepaalde modi maakt een FileStream-object mappen wanneer bestanden worden geopend. Ontbrekende mappen worden gemaakt wanneer u een FileStream-instantie instantieert met de parameter `fileMode` van de `FileStream()`-constructor ingesteld op `FileMode.APPEND` of `FileMode.WRITE`. Zie "Workflow voor het lezen van en schrijven naar bestanden" op pagina 728 voor meer informatie.

Tijdelijke mappen maken

Adobe AIR 1.0 of hoger

De klasse File bevat de methode `createTempDirectory()`, die een map maakt in de Temp-map van het systeem, zoals in het volgende voorbeeld:

```
var temp:File = File.createTempDirectory();
```

De methode `createTempDirectory()` maakt automatisch een unieke tijdelijke map (zodat u niet zelf een nieuwe unieke locatie hoeft te bepalen).

U kunt een tijdelijke map gebruiken om voorlopig tijdelijke bestanden op te slaan die worden gebruikt voor een sessie van de toepassing. Er is ook de methode `createTempFile()` voor het maken van nieuwe unieke tijdelijke bestanden in de Temp-map van het systeem.

U kunt desgewenst de tijdelijke map verwijderen voordat de toepassing wordt gesloten, aangezien deze map *niet* automatisch wordt verwijderd op alle apparaten.

Opsommingen maken van mappen

Adobe AIR 1.0 of hoger

U kunt de methode `getDirectoryListing()` of `getDirectoryListingAsync()` van een File-object gebruiken om een array op te halen van File-objecten die bestanden en submappen aanwijzen in een map.

De volgende code geeft bijvoorbeeld een lijst weer van de inhoud van de documentenmap van de gebruiker (zonder submappen te doorzoeken):

```
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);
}
```

Als u de asynchrone versie van de methode gebruikt, heeft het gebeurtenisobject `directoryListing` de eigenschap `files`, die bestaat uit de array van File-objecten die betrekking hebben op de mappen:

```
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);
    }
}
```

Mappen kopiëren en verplaatsen

Adobe AIR 1.0 of hoger

U kunt een map kopiëren of verplaatsen met behulp van dezelfde methoden die u gebruikt om een bestand te kopiëren of te verplaatsen. Met de volgende code kopieert u bijvoorbeeld een map synchroon:

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

Wanneer u "true" opgeeft voor de parameter `overwrite` van de methode `copyTo()`, worden alle bestanden en mappen in een bestaande doelmap verwijderd en vervangen door de bestanden en mappen uit de bronmap (zelfs als het doelbestand niet bestaat in de bronmap).

De map die u opgeeft bij de parameter `newLocation` van de methode `copyTo()`, definieert het pad naar de resulterende map. Deze parameter definieert *niet* de *bovenliggende* map die de resulterende map zal bevatten.

Zie "Bestanden kopiëren en verplaatsen" op pagina 724 voor meer informatie.

De inhoud van mappen verwijderen

Adobe AIR 1.0 of hoger

De klasse File bevat de methoden `deleteDirectory()` en `deleteDirectoryAsync()`. Deze methoden verwijderen mappen, de eerste op synchrone wijze, de tweede op asynchrone wijze (zie "Basisinformatie over AIR-bestanden" op pagina 705). Beide methoden bevatten de parameter `deleteDirectoryContents` (die een Booleaanse waarde heeft). Wanneer deze parameter wordt ingesteld op `true` (de standaardwaarde is `false`), verwijdert de oproep van deze methode mappen die niet leeg zijn. Bij gebruik van de standaardwaarde worden alleen lege mappen verwijderd.

De volgende code verwijdert bijvoorbeeld synchroon de submap AIR Test van de documentenmap van de gebruiker:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

De volgende code verwijdert asynchroon de submap AIR Test van de documentenmap van de gebruiker:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Ook beschikbaar zijn de methoden `moveToTrash()` en `moveToTrashAsync()`, die u kunt gebruiken om een map te verplaatsen naar de systeemprullenbak. Zie "Bestanden verplaatsen naar de prullenbak" op pagina 726 voor meer informatie.

Werken met bestanden

Adobe AIR 1.0 of hoger

Met behulp van de AIR API voor het bestandssysteem kunt u basismogelijkheden voor de interactie tussen bestanden toevoegen aan uw toepassingen. U kunt bijvoorbeeld lezen van en schrijven naar bestanden, bestanden kopiëren en verwijderen enzovoort. Aangezien uw toepassingen toegang kunnen krijgen tot het lokale bestandssysteem, is het aan te raden "Beveiliging in AIR" op pagina 1144 te raadplegen als u dat nog niet hebt gedaan.

Opmerking: *U kunt een bestandstype koppelen aan een AIR-toepassing (zodat de toepassing wordt geopend wanneer u dubbelklikt op het bestand). Zie "Bestandskoppelingen beheren" op pagina 937 voor meer informatie.*

Bestandsinformatie ophalen

Adobe AIR 1.0 of hoger

De klasse File bevat de volgende eigenschappen die informatie verstrekken over een bestand dat of een map die wordt aangewezen door een File-object:

File-eigenschap	Beschrijving
creationDate	De aanmaakdatum van het bestand op de lokale schijf.
creator	Verouderd - gebruik de eigenschap `extension`. (Deze eigenschap rapporteert het Macintosh-creatortype van het bestand; dit wordt alleen gebruikt in Mac OS-versies voorafgaand aan Mac OS X.)
downloaded	(AIR 2 en hoger) Geeft aan of het referentiebestand of -map gedownload (van het internet) is of niet. Deze eigenschap heeft alleen betekenis bij besturingssystemen waarop bestanden kunnen worden gemarkeerd als gedownload: • Windows XP service pack 2 en later en op Windows Vista • Mac OS 10.5 en later
exists	Geeft aan of het bestand of de map waarnaar wordt verwezen, bestaat.
extension	De bestandsextensie, die bestaat uit het deel van de naam na (en niet inclusief) de laatste punt ("."). Als de bestandsnaam geen punt bevat, is de extensie `null`.
icon	Een Icon-object dat de pictogrammen bevat die zijn gedefinieerd voor het bestand.
isDirectory	Geeft aan of de File-objectverwijzing betrekking heeft op een map.
modificationDate	De datum waarop het bestand of de map op de lokale schijf voor het laatst is gewijzigd.
name	De naam van het bestand of de map (inclusief de eventuele bestandsextensie) op de lokale schijf.
nativePath	Het volledige pad in de representatie van het hostbesturingssysteem. Zie "Paden van File-objecten" op pagina 706.
parent	De map die de map of het bestand bevat dat wordt vertegenwoordigd door het File-object. Deze eigenschap is `null` als het File-object verwijst naar een bestand of map in de root van het bestandssysteem.
size	De grootte van het bestand op de lokale schijf (in bytes).
type	Verouderd - gebruik de eigenschap `extension`. (Op de Macintosh is deze eigenschap het bestandstype van vier tekens; dit wordt alleen gebruikt in Mac OS-versies voorafgaand aan Mac OS X.)
url	De URL voor het bestand of de map. Zie "Paden van File-objecten" op pagina 706.

Zie het onderwerp over de klasse File in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over deze eigenschappen.

Bestanden kopiëren en verplaatsen

Adobe AIR 1.0 of hoger

De klasse File bevat twee methoden voor het kopiëren van bestanden of mappen: `copyTo()` en `copyToAsync()`. De klasse File bevat twee methoden voor het verplaatsen van bestanden of mappen: `moveTo()` en `moveToAsync()`. De methoden `copyTo()` en `moveTo()` werken synchroon, de methoden `copyToAsync()` en `moveToAsync()` asynchroon (zie "Basisinformatie over AIR-bestanden" op pagina 705).

Als u een bestand wilt kopiëren of verplaatsen, stelt u twee File-objecten in. Een van deze objecten wijst het bestand aan dat u wilt kopiëren of verplaatsen. Dit is het object dat de kopieer- of verplaatsingsmethode oproept. Het andere object wijst het doelpad (resultaat) aan.

Met de volgende code kopieert u het bestand test.txt van de submap AIR Test van de documentenmap van de gebruiker naar het bestand copy.txt in dezelfde map:

```
var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

In dit voorbeeld is de waarde van de parameter `overwrite` van de methode `copyTo()` (de tweede parameter) ingesteld op `true`. Door `overwrite` in te stellen op `true`, wordt een bestaand doelbestand overgeschreven. Deze parameter is optioneel. Als u de parameter instelt op `false` (de standaardwaarde), verzendt de bewerking de gebeurtenis IOErrorEvent als het doelbestand al bestaat. Het bestand wordt dan niet gekopieerd.

De "Async"-versies van de kopieer- en verplaatsingsmethoden werken asynchroon. Gebruik de methode `addEventListener()` om te controleren of de taak al is voltooid, en of er eventuele fouten zijn opgetreden, bijvoorbeeld met de volgende code:

```
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.");
}
```

De klasse File bevat ook de methoden `File.moveToTrash()` en `File.moveToTrashAsync()`, die een bestand of map naar de systeemprullenbak verplaatsen.

Bestanden verwijderen

Adobe AIR 1.0 of hoger

De klasse File bevat de methoden `deleteFile()` en `deleteFileAsync()`. Deze methoden verwijderen bestanden, de eerste op synchrone wijze, de tweede op asynchrone wijze (zie "Basisinformatie over AIR-bestanden" op pagina 705).

De volgende code verwijdert bijvoorbeeld synchroon het bestand test.txt uit de documentenmap van de gebruiker:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

De volgende code verwijdert asynchroon het bestand test.txt uit de documentenmap van de gebruiker:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Ook beschikbaar zijn de methoden `moveToTrash()` en `moveToTrashAsync()`, die u kunt gebruiken om een bestand of map te verplaatsen naar de systeemprullenbak. Zie "Bestanden verplaatsen naar de prullenbak" op pagina 726 voor meer informatie.

Bestanden verplaatsen naar de prullenbak

Adobe AIR 1.0 of hoger

De klasse File bevat de methoden `moveToTrash()` en `moveToTrashAsync()`. Deze methoden sturen een bestand of map naar de systeemprullenbak, de eerste synchroon, de tweede asynchroon (zie "Basisinformatie over AIR-bestanden" op pagina 705).

De volgende code verplaatst bijvoorbeeld synchroon het bestand test.txt uit de documentenmap van de gebruiker naar de systeemprullenbak:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

Opmerking: *Als een besturingssysteem geen ondersteuning biedt voor het concept van een prullenbak waarbij bestanden kunnen worden hersteld, worden de bestanden direct verwijderd.*

Tijdelijke bestanden maken

Adobe AIR 1.0 of hoger

De klasse File bevat de methode `createTempFile()`, die een bestand maakt in de Temp-map van het systeem, zoals in het volgende voorbeeld:

```
var temp:File = File.createTempFile();
```

De methode `createTempFile()` maakt automatisch een uniek tijdelijk bestand (zodat u niet zelf een nieuwe unieke locatie hoeft te bepalen).

U kunt een tijdelijk bestand gebruiken om tijdelijk informatie op te slaan die wordt gebruikt in een sessie van de toepassing. Er is ook de methode `createTempDirectory()` voor het maken van een unieke tijdelijke map in de Temp-map van het systeem.

U kunt desgewenst het tijdelijke bestand verwijderen voordat de toepassing wordt gesloten, aangezien dit bestand *niet* automatisch wordt verwijderd op alle apparaten.

Werken met opslagvolumes

Adobe AIR 2 of hoger

In AIR 2 kunt u nagaan wanneer massaopslagvolumes gekoppeld of ontkoppeld zijn. De StorageVolumeInfo-klasse definieert een `storageVolumeInfo`-singletonobject. Het object `StorageVolumeInfo.storageVolumeInfo` verzendt een `storageVolumeMount`-gebeurtenis wanneer een opslagvolume wordt gekoppeld. Er wordt een `storageVolumeUnmount`-gebeurtenis verstuurd wanneer een volume wordt ontkoppeld. Deze gebeurtenissen worden gedefinieerd door de klasse StorageVolumeChangeEvent.

Opmerking: *Bij moderne Linux-distributies verzendt het StorageVolumeInfo-object alleen `storageVolumeMount`- en `storageVolumeUnmount`-gebeurtenissen voor fysieke apparaten en netwerkstations die op specifieke locaties zijn geïnstalleerd.*

De eigenschap `storageVolume` van de StorageVolumeChangeEvent-klasse is een StorageVolume-object. De StorageVolume-klasse definieert basiseigenschappen van het opslagvolume:

- `station`—De stationsletter van het volume in Windows (`null` bij andere besturingssystemen)
- `fileSystemType` - Het type bestandssysteem van het opslagvolume (zoals FAT, NTFS, HFS of UFS).
- `isRemoveable` - Of een volume verwijderbaar is (`true`) of niet (`false`).
- `isWritable` - Geeft aan of een volume beschrijfbaar is (`true`) of niet (`false`).
- `name` - De naam van het volume.
- `rootDirectory` - Een File-object dat correspondeert met de hoofdmap van het volume.

De klasse StorageVolumeChangeEvent bevat eveneens de eigenschap `rootDirectory`. De eigenschap `rootDirectory` is een File-object met een verwijzing naar de hoofdmap van het opslagvolume dat gekoppeld of ontkoppeld is.

In geval van ontkoppelde volumes wordt de eigenschap `storageVolume` van het StorageVolumeChangeEvent-object niet gedefinieerd (`null`). U hebt echter wel toegang tot de eigenschap `rootDirectory` van de gebeurtenis.

De volgende code retourneert de naam en het bestandspad van een volume wanneer dit wordt gekoppeld:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_MOUNT, onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

De volgende code retourneert de naam en het bestandspad van een volume wanneer dit wordt ontkoppeld:

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_UNMOUNT, onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

Het object `StorageVolumeInfo.storageVolumeInfo` bevat een `getStorageVolumes()`-methode. Deze methode retourneert een vector van StorageVolume-objecten die corresponderen met de opslagvolumes die op dat moment gekoppeld zijn. De volgende code laat zien hoe u een lijst kunt weergeven met namen en hoofdmappen van alle gekoppelde opslagvolumes:

```
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);
}
```

Opmerking: *Bij moderne Linux-distributies retourneert de methode `getStorageVolumes()` objecten die overeenkomen met fysieke apparaten en netwerkstations die op specifieke locaties zijn geïnstalleerd.*

De methode `File.getRootDirectories()` retourneert een lijst met hoofdmappen (zie "Verwijzen naar het hoofdbestandssysteem" op pagina 711). StorageVolume-objecten (opgesomd door de methode `StorageVolumeInfo.getStorageVolumes()`) geven echter meer informatie over de opslagvolumes.

Met de eigenschap `spaceAvailable` van de eigenschap `rootDirectory` van een StorageVolume-object kunt u erachter komen hoeveel ruimte er op een opslagvolume beschikbaar is. (Zie "De beschikbare ruimte op een volume bepalen" op pagina 719.)

Meer Help-onderwerpen

StorageVolume

StorageVolumeInfo

Lezen van en schrijven naar bestanden

Adobe AIR 1.0 of hoger

Met de klasse FileStream kunnen AIR-toepassingen lezen en schrijven naar het bestandssysteem.

Workflow voor het lezen van en schrijven naar bestanden

Adobe AIR 1.0 of hoger

De workflow voor het lezen van en schrijven naar bestanden is als volgt.

Initialiseer een File-object dat het pad aanwijst.
Het File-object representeert het pad van het bestand waarmee u wilt werken (of een bestand dat u later zult maken).

```
 var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

In dit voorbeeld worden de eigenschap `File.documentsDirectory` en de methode `resolvePath()` van een File-object gebruikt om het File-object te initialiseren. Er zijn echter allerlei andere manieren om een File-object een bestand te laten aanwijzen. Zie "File-objecten een bestand laten aanwijzen" op pagina 713 voor meer informatie.

Initialiseer een FileStream-object.

Roep de methode open() of openAsync() van het FileStream-object op.
Welke methode u oproept, hangt ervan af of u het bestand wilt openen voor synchrone dan wel asynchrone bewerkingen. Gebruik het File-object als parameter `file` van de methode open. Bij de parameter `fileMode` geeft u een constante uit de klasse FileMode op die bepaalt op welke manier u het bestand wilt gebruiken.

De volgende code initialiseert bijvoorbeeld een FileStream-object dat wordt gebruikt om een bestand te maken en eventuele bestaande gegevens te overschrijven:

```
 var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

Zie "FileStream-objecten initialiseren en bestanden openen en sluiten" op pagina 730 en "FileStream-modi voor openen" op pagina 730 voor meer informatie.

Als u het bestand asynchroon hebt geopend (met de methode openAsync()), moet u gebeurtenislisteners toevoegen en instellen voor het FileStream-object.
Deze gebeurtenislistenermethoden reageren op gebeurtenissen die in verschillende situaties door het object FileStream worden verzonden. Het gaat hier om situaties die zich voordoen als een bestand wordt gelezen, als er I/O-fouten optreden of als het schrijven van gegevens helemaal is voltooid.

Zie "Asynchroon programmeren en de gebeurtenissen die worden gegenereerd door een FileStream-object dat asynchroon is geopend" op pagina 734 voor meer informatie.

Neem zonodig code op voor het lezen en schrijven van gegevens.
Er zijn heel veel methoden uit de klasse FileStream die betrekking hebben op lezen en schrijven. (Ze beginnen allemaal met "read" of "write".) De methode die u kiest om gegevens te lezen of te schrijven, is afhankelijk van de indeling van de gegevens in het doelbestand.

Als het doelbestand bijvoorbeeld UTF-gecodeerde tekst bevat, kunt u de methoden `readUTFBytes()` en `writeUTFBytes()` gebruiken. Als u de gegevens wilt verwerken als bytearrays, kunt u de methoden `readByte()`, `readBytes()`, `writeByte()` en `writeBytes()` gebruiken. Zie "Gegevensindelingen, en de methode voor lezen en schrijven kiezen" op pagina 735 voor meer informatie.

Als u het bestand asynchroon hebt geopend, moet u er zeker van zijn dat er voldoende gegevens beschikbaar zijn voordat u een leesmethode oproept. Zie "De leesbuffer en de eigenschap bytesAvailable van een FileStream-object" op pagina 733 voor meer informatie.

Voordat u gegevens naar een bestand schrijft, kunt u eventueel de hoeveelheid beschikbare schijfruimte nagaan door de eigenschap spaceAvailable van het File-object te controleren. Zie "De beschikbare ruimte op een volume bepalen" op pagina 719 voor meer informatie.

Roep de methode close() van het FileStream-object op wanneer u klaar bent met het bestand.
Door het aanroepen van de methode close() wordt het bestand beschikbaar voor andere toepassingen.

Zie "FileStream-objecten initialiseren en bestanden openen en sluiten" op pagina 730 voor meer informatie.

Als u een voorbeeldtoepassing wilt zien die van de klasse FileStream gebruikmaakt om bestanden te lezen en te schrijven, raadpleegt u de volgende artikelen in het Adobe AIR Developer Center:

- Een editor voor tekstbestanden maken
- Een editor voor tekstbestanden maken
- Lezen van en schrijven naar een XML-voorkeurenbestand

Werken met FileStream-objecten

Adobe AIR 1.0 of hoger

De klasse FileStream definieert methoden voor het openen, lezen en schrijven van bestanden.

FileStream-modi voor openen

Adobe AIR 1.0 of hoger

De methoden `open()` en `openAsync()` van een FileStream-object bevatten elk de parameter `fileMode`, die bepaalde eigenschappen van een bestandsstroom definieert, inclusief de volgende:

- de mogelijkheid om van het bestand te lezen;
- de mogelijkheid om naar het bestand te schrijven;
- of gegevens bij het schrijven altijd worden toegevoegd voorbij het einde van het bestand (append);
- wat er moet gebeuren als het bestand niet bestaat (en als de bovenliggende mappen niet bestaan).

Hieronder worden de verschillende bestandsmodi beschreven (die u kunt opgeven als de parameter `fileMode` van de methoden `open()` en `openAsync()`):

FileMode-type	Beschrijving
FileMode.READ	Geeft aan dat het bestand alleen open is voor lezen.
FileMode.WRITE	Geeft aan dat het bestand open is voor schrijven. Als het bestand niet bestaat, wordt het gemaakt wanneer het FileStream-object wordt geopend. Als het bestand wel bestaat, worden eventuele bestaande gegevens verwijderd.
FileMode.APPEND	Geeft aan dat het bestand open is voor het toevoegen van gegevens. Als het bestand niet bestaat, wordt het gemaakt. Als het bestand wel bestaat, worden bestaande gegevens niet overschreven. Het schrijven begint aan het einde van het bestand.
FileMode.UPDATE	Geeft aan dat het bestand open is voor lezen en schrijven. Als het bestand niet bestaat, wordt het gemaakt. Selecteer deze modus voor willekeurige lees-/schrijftoegang tot het bestand. U kunt gegevens lezen vanuit een willekeurige positie in het bestand. Wanneer u schrijft naar het bestand, worden bestaande bytes alleen overschreven door de geschreven bytes (alle overige bytes blijven ongewijzigd).

FileStream-objecten initialiseren en bestanden openen en sluiten

Adobe AIR 1.0 of hoger

Wanneer u een FileStream-object opent, stelt u het beschikbaar om gegevens in een bestand te lezen en ernaar te schrijven. U opent een FileStream-object door een File-object door te geven aan de methode `open()` of `openAsync()` van het FileStream-object:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

De parameter `fileMode` (de tweede parameter van de methoden `open()` en `openAsync()`) geeft de modus op waarin het bestand moet worden geopend: voor lezen, schrijven, toevoegen (append) of bijwerken (update). Zie de vorige sectie, "FileStream-modi voor openen" op pagina 730, voor meer informatie.

Als u de methode `openAsync()` gebruikt om het bestand te openen voor asynchrone bestandsbewerkingen, moet u gebeurtenislisteners instellen voor de afhandeling van de asynchrone gebeurtenissen:

```
 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 {
    // ...
}
```

Het bestand wordt geopend voor synchrone of asynchrone bewerkingen, afhankelijk van de methode die u gebruikt: `open()` of `openAsync()`. Zie "Basisinformatie over AIR-bestanden" op pagina 705 voor meer informatie.

Als u de parameter `fileMode` instelt op `FileMode.READ` of `FileMode.UPDATE` in de openmethode van het FileStream-object, worden gegevens in de leesbuffer ingelezen zodra u het FileStream-object opent. Zie "De leesbuffer en de eigenschap bytesAvailable van een FileStream-object" op pagina 733 voor meer informatie.

U kunt de methode `close()` van een FileStream-object oproepen om het bijbehorende bestand te sluiten, zodat dit beschikbaar wordt voor gebruik door andere toepassingen.

De eigenschap position van een FileStream-object

Adobe AIR 1.0 of hoger

De eigenschap `position` van een FileStream-object bepaalt waar gegevens worden gelezen of geschreven bij de volgende lees- of schrijfmethode.

Voordat u een lees- of schrijfbewerking uitvoert, stelt u de eigenschap `position` in op een geldige positie in het bestand.

De volgende code schrijft bijvoorbeeld de tekenreeks `"hello"` (in UTF-codering) naar positie 8 in het bestand:

```
 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");
```

Wanneer u een FileStream-object voor het eerst opent, is de eigenschap `position` ingesteld op 0.

Voordat een leesbewerking wordt uitgevoerd, moet de waarde van `position` minimaal 0 zijn en minder dan het aantal bytes in het bestand (de bestaande posities in het bestand).

De waarde van de eigenschap `position` wordt alleen gewijzigd onder de volgende omstandigheden:

- wanneer u de eigenschap `position` expliciet instelt;
- wanneer u een leesmethode oproept;

- wanneer u een schrijfmethode oproept.

Wanneer u een lees- of schrijfmethode van een FileStream-object oproept, wordt de waarde van de eigenschap `position` onmiddellijk verhoogd met het aantal bytes dat u leest of schrijft. Afhankelijk van de leesmethode die u gebruikt, wordt de waarde van de eigenschap `position` verhoogd met het aantal bytes dat u opgeeft om te lezen, of met het beschikbare aantal bytes. Wanneer u vervolgens een lees- of schrijfmethode oproept, begint deze te lezen of te schrijven bij de nieuwe positie.

```
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
```

Hiervoor geldt echter een uitzondering: bij een FileStream die wordt geopend in de toevoegmodus (append), wordt de waarde van de eigenschap `position` niet gewijzigd na het oproepen van een schrijfmethode. (In de toevoegmodus worden gegevens altijd aan het einde van het bestand geschreven, onafhankelijk van de waarde van de eigenschap `position`.)

Bij een bestand dat is geopend voor asynchrone bewerkingen, wordt de schrijfbewerking pas voltooid wanneer de volgende regel code is uitgevoerd. U kunt echter meerdere asynchrone methoden na elkaar oproepen; de runtime voert ze dan in de juiste volgorde uit:

```
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.");
}
```

De trace-uitvoer voor de code in dit voorbeeld is:

```
started.
finished.
```

U *kunt* de waarde van `position` opgeven onmiddellijk nadat u een lees- of schrijfmethode hebt opgeroepen (of op een willekeurig ander moment); de volgende lees- of schrijfbewerking vindt plaats vanaf die positie. Bij de volgende code wordt de eigenschap `position` bijvoorbeeld ingesteld onmiddellijk na het oproepen van de bewerking `writeBytes()`; `position` wordt ingesteld op die waarde (300), zelfs nadat de schrijfbewerking is voltooid:

```
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
```

De leesbuffer en de eigenschap bytesAvailable van een FileStream-object

Adobe AIR 1.0 of hoger

Wanneer een FileStream-object met leesmogelijkheden (een object waarbij de parameter `fileMode` van de methode `open()` of `openAsync()` is ingesteld op `READ` of `UPDATE`) wordt geopend, slaat de runtime de gegevens op in een interne buffer. Het FileStream-object begint gegevens in de buffer in te lezen zodra u het bestand opent (door de methode `open()` of `openAsync()` van het FileStream-object op te roepen).

Voor een bestand dat is geopend voor synchrone bewerkingen (met behulp van de methode `open()`), kunt u de aanwijzer `position` altijd instellen op een willekeurige geldige positie (binnen de grenzen van het bestand) en beginnen met het inlezen van een willekeurige hoeveelheid gegevens (binnen de grenzen van het bestand), zoals aangegeven in de volgende code (waarbij ervan wordt uitgegaan dat het bestand minimaal 100 bytes omvat):

```
 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);
```

Of een bestand nu wordt geopend voor synchrone dan wel asynchrone bewerkingen, de leesmethode leest altijd uit de "beschikbare" bytes (vertegenwoordigd door de eigenschap `bytesAvalable`). Als synchroon wordt gelezen, zijn alle bytes van het bestand op elk willekeurig moment beschikbaar. Als asynchroon wordt gelezen, worden de bytes beschikbaar gesteld vanaf de positie die wordt aangegeven door de eigenschap `position` terwijl de buffer op asynchrone wijze steeds verder wordt gevuld (gesignaleerd door `progress`-gebeurtenissen).

Voor bestanden die zijn geopend voor *synchrone* bewerkingen wordt de eigenschap `bytesAvailable` altijd zo ingesteld dat deze het aantal bytes vanaf de eigenschap `position` tot het einde van het bestand vertegenwoordigt (alle bytes in het bestand zijn altijd beschikbaar voor lezen).

Voor bestanden die zijn geopend voor *asynchrone* bewerkingen moet u ervoor zorgen dat de leesbuffer voldoende gegevens heeft verbruikt voordat een leesmethode wordt opgeroepen. Voor een bestand dat asynchroon is geopend, geldt dat naarmate de leesbewerking vordert, de gegevens uit het bestand aan de buffer worden toegevoegd, beginnende bij de waarde van `position` die werd opgegeven toen de leesbewerking werd gestart. De waarde van de eigenschap `bytesAvailable` wordt verhoogd met iedere byte die wordt gelezen. De eigenschap `bytesAvailable` geeft het aantal bytes aan dat beschikbaar is, vanaf de byte die zich bevindt op de positie die wordt aangegeven door de eigenschap `position` tot het einde van de buffer. Het FileStream-object verzendt regelmatig de gebeurtenis `progress`.

Voor een bestand dat asynchroon is geopend, verzendt het FileStream-object periodiek de gebeurtenis `progress` naarmate gegevens beschikbaar worden in de leesbuffer. De volgende code leest bijvoorbeeld gegevens in een ByteArray-object (`bytes`) terwijl dit in de buffer wordt ingelezen:

```
 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);
}
```

Voor een bestand dat asynchroon is geopend, kunnen alleen de gegevens in de leesbuffer worden gelezen. Terwijl u de gegevens leest, worden ze verwijderd uit de leesbuffer. Voor leesbewerkingen moet u ervoor zorgen dat de gegevens aanwezig zijn in de leesbuffer voordat u de leesbewerking oproept. De volgende code leest bijvoorbeeld 8000 bytes aan gegevens vanaf positie 4000 in het bestand:

```
 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");
    }
}
```

Tijdens een schrijfbewerking leest het FileStream-object geen gegevens in de leesbuffer. Wanneer een schrijfbewerking wordt voltooid (alle gegevens in de schrijfbuffer zijn naar het bestand geschreven), start het FileStream-object een nieuwe leesbuffer (ervan uitgaande dat het gekoppelde FileStream-object is geopend met leescapaciteiten) en begint het gegevens in de leesbuffer in te lezen, beginnende bij de positie die wordt aangegeven door de eigenschap `position`. De eigenschap `position` kan de positie aangeven van de byte die het laatst is geschreven, of een andere positie als de gebruiker een andere waarde voor het `position`-object na de schrijfbewerking heeft opgegeven.

Asynchroon programmeren en de gebeurtenissen die worden gegenereerd door een FileStream-object dat asynchroon is geopend

Adobe AIR 1.0 of hoger

Als een bestand asynchroon is geopend (via de methode `openAsync()`), wordt het lezen en schrijven van bestanden asynchroon uitgevoerd. Terwijl gegevens in de leesbuffer worden ingelezen en outputgegevens worden geschreven, kan andere ActionScript-code worden uitgevoerd.

Dat wil zeggen dat u moet registreren voor gebeurtenissen die worden gegenereerd door het FileStream-object dat asynchroon is geopend.

Door te registreren voor de gebeurtenis `progress` kunt u op de hoogte worden gebracht wanneer nieuwe gegevens beschikbaar worden voor lezen, zoals in de volgende code:

```
 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");
}
```

U kunt alle gegevens lezen door te registreren voor de gebeurtenis `complete`, zoals in de volgende code:

```
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");
}
```

Net zoals inputgegevens worden gebufferd om asynchroon lezen mogelijk te maken, worden gegevens die u naar een asynchrone stroom schrijft, asynchroon gebufferd en naar het bestand geschreven. Terwijl gegevens naar een bestand worden geschreven, verzendt het FileStream-object periodiek een `OutputProgressEvent`-object. Een `OutputProgressEvent`-object bevat de eigenschap `bytesPending`, die is ingesteld op het aantal bytes dat nog moet worden geschreven. U kunt registreren dat u op de hoogte moet worden gebracht van de gebeurtenis `outputProgress` terwijl deze buffer naar het bestand wordt geschreven, bijvoorbeeld om een voortgangsdialoogvenster weer te geven. In het algemeen is dit echter niet noodzakelijk. U kunt met name de methode `close()` oproepen zonder dat u zich zorgen hoeft te maken over de ongeschreven bytes. Het FileStream-object blijft gegevens schrijven en de gebeurtenis `close` wordt gegenereerd nadat de laatste byte naar het bestand is geschreven en het onderliggende bestand is gesloten.

Gegevensindelingen, en de methode voor lezen en schrijven kiezen

Adobe AIR 1.0 of hoger

Elk bestand bestaat uit een set bytes op een schijf. In ActionScript kunnen de gegevens van een bestand altijd worden weergegeven als een bytearray. De volgende code leest bijvoorbeeld de gegevens van het bestand in een ByteArray-object met de naam `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 volgende code schrijft gegevens van een ByteArray-object met de naam `bytes` naar een bestand:

```
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);
```

Het zal echter vaak gebeuren dat u de gegevens niet wilt opslaan in een ActionScript ByteArray-object. Het gegevensbestand heeft ook vaak een specifieke bestandsindeling.

De gegevens in het bestand kunnen bijvoorbeeld een tekstindeling hebben; u kunt zulke gegevens weergeven in een String-object.

Om deze reden omvat de klasse FileStream lees- en schrijfmethoden voor het lezen en schrijven van gegevens van en naar andere typen dan ByteArray-objecten. Met de methode `readMultiByte()` kunt u bijvoorbeeld gegevens lezen uit een bestand en opslaan in een tekenreeks, zoals in de volgende code:

```
 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 tweede parameter van de methode `readMultiByte()` geeft de tekstindeling aan die ActionScript gebruikt om de gegevens te interpreteren (in het voorbeeld "iso-8859-1"). Adobe AIR biedt ondersteuning voor coderingen van veelvoorkomende tekensets (zie Ondersteunde tekensets).

De klasse FileStream bevat ook de methode `readUTFBytes()`, die gegevens uit de leesbuffer in een tekenreeks inleest met gebruikmaking van de tekenset UTF-8. Aangezien tekens in de tekenset UTF-8 een variabele lengte hebben, kunt u `readUTFBytes()` niet gebruiken in een methode die reageert op de gebeurtenis `progress`, aangezien de gegevens aan het einde van de leesbuffer een onvolledig teken kunnen vertegenwoordigen. (Dit geldt ook wanneer u de methode `readMultiByte()` gebruikt met een tekencodering met variabele lengte.) Om deze reden moet de gehele set gegevens worden gelezen wanneer het FileStream-object de gebeurtenis `complete` verzendt.

Er zijn ook vergelijkbare schrijfmethoden, `writeMultiByte()` en `writeUTFBytes()`, voor het werken met String-objecten en tekstbestanden.

Ook de methoden `readUTF()` en `writeUTF()` (niet te verwarren met `readUTFBytes()` en `writeUTFBytes()`) lezen en schrijven de tekstgegevens naar een bestand, maar bij deze methoden wordt ervan uitgegaan dat de tekstgegevens worden voorafgegaan door gegevens waarmee de lengte van de tekstgegevens wordt aangegeven. Dit is niet gebruikelijk bij standaard tekstbestanden.

Bepaalde UTF-gecodeerde tekstbestanden beginnen met het teken "UTF-BOM" (byte order mark) dat de Endianness en het coderingsformaat (zoals UTF-16 of UTF-32) aangeeft.

Zie “Voorbeeld: Een XML-bestand lezen in een XML-object” op pagina 738 voor een voorbeeld van het lezen en schrijven naar een tekstbestand.

`readObject()` en `writeObject()` zijn handige manieren om gegevens op te slaan en op te halen voor complexe ActionScript-objecten. De gegevens worden gecodeerd in AMF (ActionScript Message Format). Adobe AIR, Flash Player, Flash Media Server en Flex Data Services bevatten API's voor het werken met gegevens in dit formaat.

Er zijn een paar andere lees- en schrijfmethoden (zoals `readDouble()` en `writeDouble()`). Als u deze gebruikt, moet u er zeker van zijn dat de bestandsindeling overeenkomt met de indeling van de gegevens die worden gedefinieerd door deze methoden.

Bestandsindelingen zijn vaak complexer dan eenvoudige tekstindelingen. Een MP3-bestand bevat bijvoorbeeld gecomprimeerde gegevens die alleen kunnen worden geïnterpreteerd met behulp van de decompressie- en decoderingsalgoritmen die specifiek zijn voor MP3-bestanden. MP3-bestanden kunnen ook ID3-tags bevatten die metatag-informatie over het bestand bevatten (bijvoorbeeld de titel en de artiest van een nummer). Er zijn meerdere versies van de ID3-indeling. De eenvoudigste (ID3 versie 1) wordt besproken in de sectie “Voorbeeld: Gegevens lezen en schrijven met willekeurige toegang” op pagina 739.

Andere bestandsindelingen (voor afbeeldingen, databases, toepassingsdocumenten enzovoort) hebben andere structuren. Als u met deze gegevens wilt werken in ActionScript, moet u begrijpen hoe de gegevens zijn gestructureerd.

De methoden load() en save() gebruiken

Flash Player 10 of hoger, Adobe AIR 1.5 of hoger

In Flash Player 10 zijn de methoden `load()` en `save()` toegevoegd aan de klasse FileReference. Deze methoden zijn ook beschikbaar in AIR 1.5, en de klasse File neemt de methoden over van de klasse FileReference. Deze methoden zijn ontworpen om gebruikers een veilige manier te bieden om bestandsgegevens te laden en op te slaan in Flash Player. AIR-toepassingen kunnen deze methoden echter ook gebruiken als een gemakkelijke manier om bestanden asynchroon te laden en op te slaan.

Met de volgende code slaat u bijvoorbeeld een tekenreeks op als tekstbestand:

```
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.");
}
```

De parameter `data` van de methode `save()` accepteert een tekenreeks, XML -of ByteArray-waarde. Wanneer het argument een tekenreeks of XML-waarde is, slaat deze methode het bestand op als tekstbestand met de codering UTF-8.

Wanneer dit codevoorbeeld wordt uitgevoerd, geeft de toepassing een dialoogvenster weer waarin de gebruiker de bestemming van het opgeslagen bestand moet selecteren.

De volgende code laadt een tekenreeks uit een UTF-8-gecodeerd tekstbestand:

```
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);
}
```

De klasse FileStream biedt meer functionaliteit dan die van de methoden `load()` en `save()`:

- Met de klasse FileStream kunt u gegevens zowel synchroon als asynchroon lezen en schrijven.
- Met de klasse FileStream kunt u gegevens incrementeel naar een bestand schrijven.
- Met de klasse FileStream kunt u een bestand openen voor willekeurige toegang (lezen van en schrijven naar ieder willekeurige deel van het bestand).
- Met de klasse FileStream kunt u het type bestandstoegang specificeren dat u voor het bestand hebt door de parameter `fileMode` van de methode `open()` of `openAsync()` in te stellen.
- Met de klasse FileStream kunt u gegevens opslaan in bestanden zonder dat de gebruiker het dialoogvenster Openen of Opslaan te zien krijgt.
- Bij het lezen van gegevens met de klasse FileStream kunt u rechtstreeks andere gegevenstypen dan byte arrays gebruiken.

Voorbeeld: Een XML-bestand lezen in een XML-object

Adobe AIR 1.0 of hoger

In de volgende voorbeelden ziet u hoe u kunt lezen van en schrijven naar een tekstbestand dat XML-gegevens bevat.

Om te lezen van het bestand, initialiseert u de File- en FileStream-objecten, roept u de methode `readUTFBytes()` van het FileStream-object op en converteert u de tekenreeks naar een XML-object:

```
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();
```

Om de gegevens naar het bestand te schrijven, hoeft u alleen maar de correcte File- en FileStream-objecten in te stellen en vervolgens een schrijfmethode van het FileStream-object op te roepen. Geef de tekenreeksversie van de XML-gegevens door aan de schrijfmethode zoals in de volgende code:

```
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();
```

Bij deze voorbeelden wordt gebruikgemaakt van de methoden `readUTFBytes()` en `writeUTFBytes()`, omdat er bij deze methoden van wordt uitgegaan dat de bestanden de UTF-8-indeling hebben. Als dat niet het geval is, moet u mogelijk een andere methode gebruiken (zie "Gegevensindelingen, en de methode voor lezen en schrijven kiezen" op pagina 735).

Bij de voorgaande voorbeelden worden FileStream-objecten gebruikt die zijn geopend voor synchrone bewerkingen. U kunt ook bestanden openen voor asynchrone bewerkingen (die gebruikmaken van gebeurtenislistenerfuncties om te reageren op gebeurtenissen). In de volgende voorbeeldcode ziet u hoe u een XML-bestand asynchroon kunt lezen:

```
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();
}
```

De methode `processXMLData()` wordt opgeroepen wanneer het gehele bestand in de leesbuffer is ingelezen (wanneer het FileStream-object de gebeurtenis `complete` verzendt). Deze roept de methode `readUTFBytes()` op om een tekenreeksversie van de leesgegevens te verkrijgen en maakt op basis van die tekenreeks het XML-object `prefsXML`.

Een voorbeeldtoepassing waarin deze mogelijkheden worden getoond, vindt u in Lezen van en schrijven naar een XML-voorkeurenbestand.

Voorbeeld: Gegevens lezen en schrijven met willekeurige toegang

Adobe AIR 1.0 of hoger

MP3-bestanden kunnen ID3-tags bevatten. Dit zijn secties aan het begin of einde van een bestand die metagegevens bevatten die de opname identificeren. Van de indeling van de ID3-tag bestaan verschillende versies. In dit voorbeeld ziet u hoe u kunt lezen van en schrijven naar een MP3-bestand dat de eenvoudigste ID3-indeling bevat (ID3 versie 1.0) met gebruikmaking van willekeurige toegang tot de gegevens. Dit wil zeggen dat wordt gelezen van en geschreven naar willekeurige locaties in het bestand.

Bij een MP3-bestand dat een ID3-tag van versie 1 bevat, bevinden de ID3-gegevens zich aan het einde van het bestand, in de laatste 128 bytes.

Wanneer een bestand wordt geopend voor willekeurige lees-/schrijftoegang, is het belangrijk om `FileMode.UPDATE` op te geven als de parameter `fileMode` voor de methode `open()` of `openAsync()`:

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

Hiermee kunt u zowel lezen van als schrijven naar het bestand.

Bij het openen van het bestand kunt u de aanwijzer `position` neerzetten op de positie 128 bytes vóór het einde van het bestand:

```
fileStr.position = file.size - 128;
```

Deze code stelt de eigenschap `position` in op deze locatie in het bestand, aangezien de ID3-indeling v1.0 bepaalt dat de ID3-taggegevens worden opgeslagen in de laatste 128 bytes van het bestand. Deze specificatie bevat verder het volgende:

- De eerste 3 bytes van de tag bevatten de tekenreeks `"TAG"`.
- De volgende 30 tekens bevatten de titel van de MP3-track in de vorm van een tekenreeks.
- De volgende 30 tekens bevatten de naam van de artiest in de vorm van een tekenreeks.
- De volgende 30 tekens bevatten de naam van het album in de vorm van een tekenreeks.
- De volgende 4 tekens bevatten het jaar in de vorm van een tekenreeks.
- De volgende 30 tekens bevatten het commentaar in de vorm van een tekenreeks.
- De volgende byte bevat een code die het genre van de track aangeeft.
- Alle tekstgegevens hebben de indeling ISO 8859-1.

De methode `id3TagRead()` controleert de gegevens nadat deze zijn ingelezen (na het plaatsvinden van de gebeurtenis `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);
    }
}
```

U kunt ook met willekeurige toegang naar het bestand schrijven. U kunt bijvoorbeeld de variabele `id3Title` parseren om er zeker van te zijn dat het correcte hoofdlettergebruik wordt toegepast (met behulp van methoden van de klasse String) en dan een gewijzigde tekenreeks met de naam `newTitle` naar het bestand schrijven, zoals in de volgende code:

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

Ten einde te voldoen aan de standaard voor ID3 versie 1, moet de lengte van de tekenreeks `newTitle` 30 tekens zijn, aan het einde opgevuld met de tekencode 0 (`String.fromCharCode(0)`).

Hoofdstuk 39: Lokale gegevens opslaan

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse SharedObject kunt u kleine hoeveelheden gegevens opslaan op de clientcomputer. In Adobe AIR kunt u met de klasse EncryptedLocalStore kleine hoeveelheden privacygevoelige gebruikersgegevens op de lokale computer opslaan in een AIR-toepassing.

U kunt ook bestanden lezen en schrijven op het bestandssysteem en (in Adobe AIR) lokale gegevensbestanden van de database openen. Zie voor meer informatie "Werken met het bestandssysteem" op pagina 689 en "Werken met lokale SQL-databases in AIR" op pagina 754.

Er zijn een aantal veiligheidsfactoren die betrekking hebben op gezamenlijke objecten. Zie voor meer informatie "Gezamenlijke objecten" op pagina 1142 in "Beveiliging" op pagina 1106.

Gezamenlijke objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een gezamenlijk object, soms ook wel een 'Flash-cookie' genoemd, is een gegevensbestand dat op uw computer kan worden gemaakt door de sites die u bezoekt. Gezamenlijke objecten worden meestal gebruikt om uw webervaringen te verbeteren. Zo kunt u hiermee bijvoorbeeld het uiterlijk aanpassen van een website die u vaak bezoekt.

Informatie over gezamenlijke objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gezamenlijke objecten werken als browsercookies. Met de klasse SharedObject kunt u gegevens opslaan op de lokale vaste schijf van de gebruiker. Vervolgens kunt u de gegevens in dezelfde sessie of in een latere sessie aanroepen. Toepassingen hebben uitsluitend toegang tot hun eigen SharedObject-gegevens en alleen als deze in hetzelfde domein worden uitgevoerd. De gegevens worden niet naar de server gestuurd en zijn niet toegankelijk voor andere toepassingen die andere domeinen uitvoeren, maar deze kunnen wel toegankelijk worden gemaakt voor andere toepassingen in hetzelfde domein.

Gezamenlijke objecten vergeleken met cookies

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Cookies en gezamenlijke objecten lijken veel op elkaar. De meeste webprogrammeurs weten hoe cookies werken. Daarom kan het handig zijn een vergelijking te maken tussen cookies en lokale gezamenlijke objecten.

Cookies die voldoen aan de RFC 2109-standaard hebben over het algemeen de volgende eigenschappen:

- Ze kunnen verlopen en doen dat vaak standaard aan het einde van een sessie.
- Ze kunnen op site-specifieke basis door de client worden uitgeschakeld.
- Er bestaat een limiet van 300 cookies in totaal en maximaal 20 cookies per site.
- De grootte ervan is doorgaans beperkt tot 4 kB elk

- Ze worden soms beschouwd als bedreiging van de veiligheid en worden daarom soms op de client uitgeschakeld.
- Ze worden opgeslagen op een locatie die wordt aangegeven door de clientbrowser.
- Ze worden via HTTP overgedragen van client naar server.

  Gezamenlijke objecten hebben daarentegen de volgende eigenschappen:

- Ze verlopen standaard niet.
- De grootte is standaard beperkt tot 100 kB elk.
- Er kunnen eenvoudige soorten gegevens in worden opgeslagen (zoals een rekenreeks, array en datum).
- Ze worden opgeslagen op een locatie die wordt opgegeven door de toepassing (in de basismap van de gebruiker).
- Ze worden nooit overgedragen tussen de client en de server.

Informatie over de klasse SharedObject

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse SharedObject kunt u gedeelde objecten maken en verwijderen. U kunt ook het huidige formaat bepalen van een SharedObject-object dat u gebruikt.

Gezamenlijke objecten maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een SharedObject-object wilt maken, gebruikt u de methode `SharedObject.getLocal()` met de volgende syntaxis:

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

In het volgende voorbeeld wordt een gezamenlijk object gemaakt met de naam mySO:

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

Hierbij wordt op het clientapparaat een bestand gemaakt met de naam preferences.sol.

De term *lokaal* verwijst naar de locatie van het gezamenlijke object. In dit geval slaat Adobe® Flash® Player het SharedObject-bestand lokaal op in de basismap van de client.

Wanneer u een gezamenlijk object maakt, maakt Flash Player een nieuwe map voor de toepassing en domein in de sandbox. Daarnaast wordt een *.sol-bestand gemaakt waarin de SharedObject-gegevens worden opgeslagen. De standaardlocatie van dit bestand is een submap van de basismap van de gebruiker. In de volgende tabel staan de standaardlocaties van deze map:

Besturingssysteem	Locatie
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`

Onder de map #SharedObjects bevindt zich een map met een willekeurige naam. Daaronder bevindt zich een map waarvan de naam overeenkomt met de hostnaam, gevolgd door het pad naar de toepassing en tot slot het *.sol-bestand.

Als u bijvoorbeeld vraagt om een toepassing met de naam MyApp.swf op de lokale host en in een submap met de naam /sos, slaat Flash Player het *.sol-bestand in Windows XP op de volgende locatie op:

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

Opmerking: *Als u geen naam opgeeft in de methode* `SharedObject.getLocal()`*, geeft Flash Player het bestand de naam undefined.sol.*

Standaard kan Flash lokaal permanente SharedObject-object met een grootte van maximaal 100 kB per domein opslaan. Deze waarde kan door de gebruiker worden ingesteld. Wanneer de toepassing probeert gegevens op te slaan in een gezamenlijk object waardoor dit groter zou worden dan 100 kB, geeft Flash Player het dialoogvensters Lokale opslag weer, waarin de gebruiker meer lokale opslag voor het domein dat toegang vraagt kan toestaan of weigeren.

Pad opgeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de optionele parameter *pathname* geeft u een locatie op voor het SharedObject-bestand. Dit bestand moet een submap zijn van de SharedObject-map van dat domein. Als u bijvoorbeeld vraagt naar een toepassing op de lokale host en het volgende opgeeft:

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

schrijft Flash Player het SharedObject-bestand in de map /#localhost (of /localhost als de toepassing offline is). Dit is nuttig als u wilt dat meer dan één toepassing op de client toegang heeft tot hetzelfde gezamenlijke object. In dit geval zou de client twee Flex-toepassingen kunnen uitvoeren, die beide een pad opgeven naar het gezamenlijke object dat de root vormt van het domein; de client zou dan vanuit beide toepassingen toegang hebben tot hetzelfde gezamenlijke object. Als u gegevens wilt delen tussen meer dan één toepassing zonder persistentie, kunt u het object LocalConnection gebruiken.

Als u een map opgeeft die niet bestaat, maakt Flash Player geen SharedObject-bestand.

Gegevens toevoegen aan een gezamenlijk object

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt gegevens toevoegen aan een *.sol-bestand van SharedObject door gebruik te maken van de eigenschap `data` van het SharedObject-object. Gebruik de volgende syntaxis om nieuwe gegevens toe te voegen aan het gezamenlijke object:

```
sharedObject_name.data.variable = value;
```

In het volgende voorbeeld worden de eigenschappen `userName`, `itemNumbers` en `adminPrivileges` en hun waarden toegevoegd aan een 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;
```

Wanneer u waarden hebt toegewezen aan de eigenschap `data`, moet u Flash Player opdragen die waarden naar het SharedObject-bestand te schrijven. Om Flash Player te dwingen de waarden naar het SharedObject-bestand te schrijven, gebruikt u de methode `SharedObject.flush()` als volgt:

```
mySO.flush();
```

Als u de methode `SharedObject.flush()` niet aanroept, schrijft Flash Player de waarden naar het bestand wanneer de toepassing wordt afgesloten. Dit biedt de gebruiker echter niet de gelegenheid om de beschikbare ruimte te vergroten die Flash Player heeft om de gegevens op te slaan als de gegevens groter zijn dan de standaardinstellingen. Het is daarom een goed idee om `SharedObject.flush()` aan te roepen.

Als u de methode `flush()` gebruikt om gezamenlijke objecten naar de vaste schijf van de gebruiker te schrijven, moet u controleren of de gebruiker lokale opslag expliciet heeft uitgeschakeld via Settings Manager van Flash Player (www.macromedia.com/support/documentation/nl/flashplayer/help/settings_manager07.html), zoals in het volgende voorbeeld:

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

Objecten opslaan in gezamenlijke objecten

U kunt eenvoudige objecten, zoals arrays of tekenreeksen opslaan in de eigenschap `data` van een SharedObject.

In het volgende voorbeeld definieert de klasse ActionScript methoden waarmee de interactie met het gezamenlijke object wordt geregeld. Met deze methoden kan de gebruiker objecten aan het gezamenlijke object toevoegen of eruit verwijderen. In deze klasse wordt een ArrayCollection opgeslagen die eenvoudige objecten bevat.

```
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();
        }
    }

}
```

Met de volgende Flex-toepassing wordt een instantie van de klasse ActionScript gemaakt voor elk type gezamenlijk object dat nodig is. Vervolgens worden methoden aangeroepen op die klasse wanneer de gebruiker blogs of site-URL's toevoegt of verwijdert.

```
<?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>
```

Objecten met typen opslaan in gezamenlijke objecten

U kunt instanties van ActionScript met type opslaan in gezamenlijke objecten. Dit doet u door de methode `flash.net.registerClassAlias()` aan te roepen om de klasse te registreren. Als u een instantie van de klasse maakt en deze opslaat in het gegevenslid van het gezamenlijke object en het object later uitleest, krijgt u een instantie met een type. Standaard ondersteunt de SharedObject-eigenschap `objectEncoding` AMF3-codering en wordt de opgeslagen instantie uit het SharedObject-object uitgepakt; de opgeslagen instantie houdt het type dat u hebt opgegeven toen u de methode `registerClassAlias()` aanriep.

(alleen iOS) Back-up in de cloud verhinderen voor gedeelde objecten

Adobe AIR 3.7 en later, alleen iOS

U kunt de `SharedObject.preventBackup`-eigenschap instellen om te bepalen of lokaal gedeelde objecten in de back-up van de iOS-cloudback-upservice worden geplaatst. Dit is verplicht door Apple voor inhoud die opnieuw kan worden gegenereerd of gedownload, maar die vereist is voor een goede werking van uw toepassing tijdens offlinegebruik.

Meerdere gezamenlijke objecten maken

U kunt meerdere gezamenlijke objecten maken voor dezelfde Flex-toepassing. Hiertoe wijst u aan elk hiervan een andere instantienaam toe, zoals aangegeven in het volgende voorbeeld:

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

Hiermee maakt u een preferences.sol-bestand en een history.sol-bestand aan in de lokale map van de Flex-toepassing.

Een beveiligd SharedObject maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een lokaal of extern SharedObject maakt met behulp van `getLocal()` of `getRemote()`, bepaalt een optionele parameter met de naam `secure` of toegang tot dit gezamenlijke object is beperkt tot SWF-bestanden die worden aangeboden via een HTTPS-verbinding. Als deze parameter wordt ingesteld op `true` en het SWF-bestand wordt aangeboden via HTTPS, wordt in Flash Player een nieuw beveiligd gezamenlijk object gemaakt of wordt een verwijzing naar een bestaand beveiligd gezamenlijk object opgehaald. Alleen SWF-bestanden die via HTTPS worden geleverd en `SharedObject.getLocal()` aanroepen met de parameter secure die is ingesteld op `true`, hebben lees- en schrijftoegang tot dit beveiligde gezamenlijke object. Als deze parameter wordt ingesteld op `false` en het SWF-bestand wordt aangeboden via HTTPS, wordt in Flash Player een nieuw gezamenlijk object gemaakt of wordt een verwijzing naar een bestaand gezamenlijk object opgehaald.

SWF-bestanden die via niet-HTTPS-verbindingen worden geleverd, hebben lees- en schrijftoegang tot dit gezamenlijke object. Als uw SWF-bestand wordt geleverd via een niet-HTTPS-verbinding en u deze parameter probeert in te stellen op `true`, mislukt het maken van een nieuw gezamenlijk object (of kan er geen toegang worden verkregen tot een eerder gemaakt beveiligd gezamenlijk object). Er wordt een fout gegenereerd en het gezamenlijke object wordt ingesteld op `null`. Als u probeert het volgende fragment uit te voeren vanuit een niet-HTTPS-verbinding, genereert de methode `SharedObject.getLocal()` een fout:

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

Ongeacht de waarde van deze parameter tellen de gemaakte gezamenlijke objecten mee voor de toegestane totale hoeveelheid schijfruimte voor een domein.

De inhoud van een gezamenlijk object weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Waarden worden in gezamenlijke objecten opgeslagen in de eigenschap `data`. U kunt alle waarden binnen een instantie van een gezamenlijk object doorlopen met een `for..in`-lus, zoals in het volgende voorbeeld wordt getoond:

```
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]);
}
```

Gezamenlijke objecten vernietigen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een SharedObject op de client wilt vernietigen, kan dit met de methode `SharedObject.clear()`. Hiermee vernietigt u geen mappen in het standaardpad van de gezamenlijke objecten van de toepassing.

In het volgende voorbeeld wordt het SharedObject-bestand op de client vernietigd:

```
public function destroySharedObject():void {
    mySO.clear();
}
```

SharedObject-voorbeeld

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het volgende voorbeeld wordt duidelijk dat u eenvoudige objecten, zoals een Date-object, kunt opslaan in een SharedObject-object, zonder dat u de objecten handmatig moet serialiseren en deserialiseren.

In het volgende voorbeeld wordt u eerst als nieuwe bezoeker verwelkomd. Wanneer u op Afmelden klikt, slaat de toepassing de huidige datum op in een gezamenlijk object. De volgende keer dat u de toepassing start of de pagina vernieuwt, verwelkomt de toepassing u opnieuw en laat zien wanneer u zich hebt afgemeld.

Als u de toepassing in actie wilt zien, start u de toepassing, klikt u op Afmelden en vernieuwt u vervolgens de pagina. De toepassing geeft de datum en de tijd weer van het moment waarop u op de knop Afmelden klikte bij uw vorige bezoek. U kunt de opgeslagen informatie op elk moment verwijderen door op de knop LSO verwijderen te klikken.

```
<?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>
```

Lokale gegevens gecodeerd opslaan

De ELS-klasse (EncryptedLocalStore) biedt een manier om lokale gegevens gecodeerd op te slaan die u kunt gebruiken als een kleine cache voor de privégegevens van een toepassing. ELS-gegevens kunnen niet worden gedeeld tussen toepassingen. De bedoeling van ELS is het mogelijk maken dat een toepassing opnieuw gemaakte items, zoals aanmeldingsgegevens en andere privégegevens, gemakkelijk kan opslaan. ELS-gegevens moeten niet worden beschouwd als permanente gegevens, zoals hieronder beschreven in "Beperkingen van de gecodeerde lokale opslag" en "Tips en trucs".

Opmerking: *Naast de gecodeerde lokale opslag biedt AIR ook codering voor inhoud die is opgeslagen in SQL-databases.* *Zie voor meer informatie "Codering gebruiken met SQL-databases" op pagina 800.*

In de gecodeerde lokale opslag kunt u cache-informatie opslaan die moet worden beveiligd, zoals aanmeldingsgegevens voor webservices. Deze gecodeerde lokale opslag, ofwel ELS (Encrypted Local Store), is geschikt voor het opslaan van informatie die andere gebruikers niet mogen zien. De gegevens worden dan echter niet beveiligd tegen andere processen die onder dezelfde gebruikersaccount worden uitgevoerd. De ELS is dus niet geschikt voor het beveiligen van geheime toepassingsgegevens, zoals DRM- of coderingssleutels.

Op bureaubladplatforms maakt AIR in Windows van DPAPI, in Mac OS en iOS van KeyChain en in Linux van KeyRing of KWallet gebruik om de gecodeerde lokale opslag te koppelen aan elke toepassing en gebruiker. De gecodeerde lokale opslag gebruikt AES-CBC 128-bits codering.

Op Android worden de gegevens die door de klasse EncryptedLocalStorage zijn opgeslagen niet gecodeerd. De gegevens worden in plaats daarvan beveiligd door beveiliging op gebruikersniveau die wordt verschaft door het besturingssysteem. Het Android-besturingssysteem wijst een eigen gebruikers-id toe aan elke toepassing. Toepassingen kunnen alleen hun eigen bestanden openen, plus bestanden die zijn gemaakt op openbare locaties (zoals een verwijderbare opslagkaart). Op Android-apparaten met rooting hebben toepassingen met rootingrechten WEL toegang tot de bestanden van andere toepassingen. Op een apparaat met rooting verschaft gecodeerde lokale opslag dus minder goede gegevensbeveiliging dan op een apparaat zonder rooting.

Informatie in de gecodeerde lokale opslag is alleen beschikbaar voor de inhoud van een AIR-toepassing in de beveiligingssandbox van de toepassing.

Als u een AIR-toepassing bijwerkt, hebt u ook in de bijgewerkte versie toegang tot alle bestaande gegevens in de gecodeerde lokale opslag, tenzij:

- De items zijn toegevoegd met de parameter `stronglyBound` ingesteld op `true`.
- De bestaande en bijgewerkte versie allebei ouder zijn dan AIR 1.5.3 en de update ondertekend is met een migratiehandtekening.

Beperkingen van de gecodeerde lokale opslag

De gegevens in de gecodeerde lokale opslag worden beveiligd door de verificatiegegevens van het besturingssysteem van de gebruiker. Andere entiteiten hebben alleen toegang tot de gegevens in de opslag als ze zich als de desbetreffende gebruiker kunnen aanmelden. De gegevens zijn echter niet beveiligd tegen toegang vanuit andere toepassingen die door een geverifieerde gebruiker worden uitgevoerd.

Omdat de gebruiker voor deze aanvallen geverifieerd moet zijn, zijn de privégegevens van de gebruiker nog steeds beveiligd (behalve als de gebruikersaccount is gecompromitteerd). Gegevens die uw toepassing eventueel geheim wilt houden voor gebruikers, zoals keys voor licentie- of digitale rechten, zijn echter niet beveiligd. De ELS is dus niet de geschikte locatie om zulke informatie op te slaan. Het is alleen een geschikte locatie voor het opslaan van privégegevens van een gebruiker, zoals wachtwoorden.

De gegevens in de ELS kunnen om verschillende redenen verloren gaan. Een gebruiker kan de toepassing bijvoorbeeld verwijderen en daarmee het gecodeerde bestand. Het is ook mogelijk dat de uitgever-id wordt gewijzigd als resultaat van een update. De lokale gegevensopslag moet dus worden beschouwd als een persoonlijk cachegeheugen, niet als een permanente gegevensopslag.

De parameter `stronglyBound` is afgekeurd en mag niet worden ingesteld op `true`. Het instellen van de parameter op `true` verschaft uw gegevens geen extra beveiliging. Tegelijkertijd gaat toegang tot de gegevens verloren wanneer de toepassing wordt bijgewerkt, ook als de uitgever-id niet wordt gewijzigd.

De gecodeerde lokale opslag kan langzamer werken als de opgeslagen hoeveelheid gegevens groter is dan 10 MB.

Wanneer u een AIR-toepassing verwijdert, worden de gegevens die in de gecodeerde lokale opslag zijn opgeslagen, niet verwijderd.

Tips en trucs

Tips en trucs voor het gebruik van de ELS:

- Gebruik de ELS om gevoelige gebruikersgegevens zoals wachtwoorden op te slaan (stel stronglyBound in op false).
- Gebruik de ELS niet om toepassingsgeheimen op te slaan, zoals DRM-sleutels of licentietokens..
- Verschaf uw toepassing een manier om de gegevens in de ELS opnieuw te creëren als de ELS-gegevens verloren gaan. Vraag de gebruiker bijvoorbeeld om zijn of haar aanmeldinggegevens voor de account, indien nodig, opnieuw in te voeren.
- Gebruik de parameter `stronglyBound` niet.
- Als u de parameter `stronglyBound` wel instelt op `true`, dient u opgeslagen items tijdens een update niet te verplaatsen. Het is beter de gegevens na de update opnieuw te maken.
- Sla alleen relatief kleine hoeveelheden gegevens op. Gebruik een gecodeerde AIR SQL-database voor grotere hoeveelheden gegevens.

Meer Help-onderwerpen

flash.data.EncryptedLocalStore

Gegevens toevoegen aan de gecodeerde lokale opslagplaats.

Gebruik de statische methode `setItem()` van de EncryptedLocalStore-klasse om gegevens lokaal op te slaan. De gegevens worden opgeslagen in een hashtabel, waarbij tekenreeksen worden gebruikt als sleutels en de gegevens worden opgeslagen als bytearrays.

Met de volgende code wordt bijvoorbeeld een tekenreeks opgeslagen in de gecodeerde lokale opslag:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

De derde parameter van de methode `setItem()`, de parameter `stronglyBound`, is optioneel. Als deze parameter is ingesteld op `true`, koppelt de ELS het opgeslagen item aan de digitale handtekening en onderdelen van de AIR-toepassing waarin het item wordt opgeslagen:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

Voor een item dat is opgeslagen terwijl `stronglyBound` op `true` is ingesteld, zullen volgende oproepen van `getItem()` alleen slagen als de oproepende AIR-toepassing identiek is aan de opslaande toepassing (als er geen gegevens in bestanden in de toepassingsmap zijn gewijzigd). Als de oproepende AIR-toepassing niet identiek is aan de opslaande toepassing, genereert de toepassing een uitzonderingsfout wanneer u `getItem()` oproept voor een sterk gebonden item. Als u uw toepassing bijwerkt, kan deze geen sterk gebonden gegevens lezen die eerder in de gecodeerde lokale opslag zijn geschreven. Het instellen van `stronglyBound` op `true` op mobiele apparaten wordt genegeerd, de parameter wordt altijd behandeld als `false`.

Als de parameter `stronglyBound` op `false` is ingesteld (de standaardinstelling) hoeft alleen de uitgevers-id hetzelfde te blijven, zodat de toepassing de gegevens nog steeds kan lezen. De onderdelen van de toepassing kunnen veranderen (en deze moeten door dezelfde uitgever worden ondertekend), maar het hoeven niet precies dezelfde onderdelen te zijn als in de toepassing waarin de gegevens zijn opgeslagen. Bijgewerkte toepassingen met dezelfde uitgever-id als de oorspronkelijke toepassingen hebben nog steeds toegang tot de gegevens.

Opmerking: *In de praktijk levert het instellen van* `stronglyBound` *op* `true` *geen aanvullende gegevensbeveiliging op. Een kwaadwillende gebruiker kan een toepassing not steeds aanpassen en toegang krijgen tot de items in de ELS. Bovendien zijn gegevens net zo goed beveiligd tegen externe bedreigingen van niet-gebruikers wanneer* `stronglyBound` *is ingesteld op* `true` *als op* `false`*. Daarom wordt het niet aangeraden om* `stronglyBound` *in te stellen op* `true`*.*

Gegevens openen in de gecodeerde lokale opslagplaats

Adobe AIR 1.0 of hoger

U kunt in de gecodeerde lokale opslagplaats een waarde zoeken met de `EncryptedLocalStore.getItem()`-methode, zoals in het volgende voorbeeld:

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

Gegevens verwijderen uit de gecodeerde lokale opslagplaats

Adobe AIR 1.0 of hoger

U kunt een waarde uit de gecodeerde lokale opslag verwijderen met de methode `EncryptedLocalStore.removeItem()`, zoals in het volgende voorbeeld:

```
EncryptedLocalStore.removeItem("firstName");
```

U kunt alle gegevens in de gecodeerde lokale opslag wissen door de methode `EncryptedLocalStore.reset()` op te roepen, zoals in het volgende voorbeeld:

```
EncryptedLocalStore.reset();
```

Hoofdstuk 40: Werken met lokale SQL-databases in AIR

Adobe AIR 1.0 of hoger

Adobe® AIR® biedt de mogelijkheid om lokale SQL-databases te maken er ermee te werken. De runtime bevat een SQL-database-engine met ondersteuning voor vele standaardfuncties van SQL via het open-source databasesysteem SQLite. U kunt een lokale SQL-database gebruiken om lokale, niet-vluchtige gegevens op te slaan. Deze kan bijvoorbeeld worden gebruikt voor toepassingsgegevens, gebruikersinstellingen voor de toepassing of gegevens van een ander type die u lokaal met de toepassing wilt opslaan.

Informatie over lokale SQL-databases

Adobe AIR 1.0 of hoger

Lees voor een snelle uitleg en codevoorbeelden van het gebruik van SQL-databases, de volgende snelstartartikelen in de Adobe Developer Connection:

- Working asynchronously with a local SQL database (Flex)
- Working synchronously with a local SQL database (Flex)
- Using an encrypted database (Flex)
- Working asynchronously with a local SQL database (Flash)
- Working synchronously with a local SQL database (Flash)
- Using an encrypted database (Flash)

Adobe AIR bevat een op SQL gebaseerde engine voor relationele databases die binnen de runtime wordt uitgevoerd. De gegevens worden lokaal opgeslagen in databasebestanden op de computer waarop de AIR-toepassing wordt uitgevoerd (bijvoorbeeld de vaste schijf van de computer). Aangezien de database lokaal wordt uitgevoerd en de gegevensbestanden lokaal worden opgeslagen, kan een database altijd door een AIR-toepassing worden gebruikt, ongeacht of een netwerkverbinding aanwezig is. Dit betekent dat de lokale SQL-database-engine van de runtime een handige manier is om niet-vluchtige, lokale toepassingsgegevens op te slaan, met name als u ervaring hebt met SQL- en relationele databases.

Toepassingen voor lokale SQL-databases

Adobe AIR 1.0 of hoger

De AIR-functies voor lokale SQL-databases kunnen worden gebruikt voor willekeurige toepassingen waarbij toepassingsgegevens moeten worden opgeslagen op de lokale computer van de gebruiker. Adobe AIR biedt verschillende manieren om gegevens lokaal op te slaan, waarbij elke manier andere voordelen heeft. Hieronder ziet u een aantal mogelijke toepassingen voor een lokale SQL-database in uw AIR-toepassing:

- Voor een op gegevens gebaseerde toepassing (bijvoorbeeld een adresboek) kunt u een database gebruiken om de hoofdtoepassingsgegevens op te slaan.

- Voor een op documenten gebaseerde toepassing (waarbij gebruikers documenten creëren om ze op te slaan en mogelijk te delen) kan elk document als een databasebestand worden opgeslagen op een door de gebruiker opgegeven locatie. (Houd er wel rekening mee dat als de database niet is gecodeerd, iedere AIR-toepassing in staat is het databasebestand te openen. Het is aan te raden mogelijk gevoelige documenten te coderen.)
- Voor een toepassing met netwerkverbindingen kunt u een database gebruiken om toepassingsgegevens op te slaan in een lokale cache, of om gegevens tijdelijk op te slaan wanneer er geen netwerkverbinding beschikbaar is. U kunt desgewenst een procedure creëren om de lokale database te synchroniseren met de netwerklocatie voor gegevensopslag.
- Voor alle toepassingen kunt u een database gebruiken om de toepassingsinstellingen van individuele gebruikers op te slaan, bijvoorbeeld gebruikersopties of toepassingsinformatie zoals venstergrootte en -positie.

Meer Help-onderwerpen

Christophe Coenraets: Employee Directory on AIR for Android

Raymond Camden: jQuery and AIR - Moving from web page to application

Informatie over AIR-databases en databasebestanden

Adobe AIR 1.0 of hoger

Een individuele, lokale SQL-database van Adobe AIR wordt als één bestand opgeslagen in het bestandssysteem van de computer. De runtime bevat de SQL-database-engine die het creëren en indelen van databasebestanden, en het bewerken en ophalen van gegevens in een databasebestand beheert. De runtime geeft niet aan hoe of waar de databasegegevens worden opgeslagen in het bestandssysteem - elke database bestaat uit één bestand. U geeft de locatie voor het databasebestand in het bestandssysteem op. Eén AIR-toepassing kan een of meer databases (met andere woorden aparte databasebestanden) benaderen. Aangezien de runtime elke database als één bestand opslaat in het bestandssysteem, kunt u de locatie van uw database afstemmen op het ontwerp van uw toepassing en de toegangsbeperkingen van het besturingssysteem. Iedere gebruiker kan een apart databasebestand voor zijn/haar specifieke gegevens hebben, of u kunt één databasebestand voor alle gebruikers van de toepassing op één computer plaatsen zodat de gegevens kunnen worden gedeeld. Aangezien de gegevens zich lokaal op één computer bevinden, worden de gegevens niet automatisch gedeeld met gebruikers op andere computers. De lokale SQL-database-engine biedt geen mogelijkheid om SQL-instructies uit te voeren in een externe of servergebaseerde database.

Informatie over relationele databases

Adobe AIR 1.0 of hoger

Een relationele database is een mechanisme voor het opslaan (en ophalen) van gegevens op een computer. De gegevens zijn verdeeld in tabellen: rijen geven records of items weer, en kolommen (worden ook soms “velden” genoemd) verdelen elke record in individuele waarden. Een adresboektoepassing kan bijvoorbeeld een tabel met de naam “vrienden” bevatten. Elke rij van de tabel is dan één vriend die in de database is opgeslagen. De kolommen van de tabel bevatten gegevens zoals voornaam, achternaam, geboortedatum enzovoort. Voor iedere vriendrij van de tabel slaat de database een aparte waarde voor elke kolom op.

Relationele databases zijn ontworpen voor de opslag van complexe gegevens, waarbij een item is gekoppeld of gerelateerd aan items van een ander type. In een relationele database worden gegevens met een één-op-vele-relatie (waarbij één record kan zijn gekoppeld aan meerdere records van een ander type) doorgaans verdeeld over verschillende tabellen. Als u bijvoorbeeld uw adresboektoepassing wilt gebruiken om voor iedere vriend meerdere telefoonnummers op te slaan, is dat een één-op-vele-relatie. De tabel “vrienden” bevat in dat geval alle persoonlijke gegevens van iedere vriend. Een andere tabel, “telefoonnummers”, kan dan alle telefoonnummers van alle vrienden bevatten.

In de database worden niet alleen de gegevens van vrienden en telefoonnummers opgeslagen. Elke tabel heeft ook een gegevensitem nodig om de relatie tussen de twee tabellen bij te houden, met andere woorden om individuele vriendrecords te koppelen aan de overeenkomstige telefoonnummers. Dit gegevensitem wordt een primaire sleutel genoemd, een unieke identificator die elke tabelrij verschillend maakt van alle overige rijen uit die tabel. De primaire sleutel kan een “natuurlijke sleutel” zijn, waarbij de sleutel een van de gegevensitems is die elke record uit een tabel op een natuurlijke manier uniek maken. Als u in de tabel “vrienden” bijvoorbeeld weet dat al uw vrienden een andere geboortedatum hebben, kunt u de kolom voor de geboortedatum instellen als primaire sleutel (een natuurlijke sleutel) voor de tabel “vrienden”. Als er geen natuurlijke sleutel is, maakt u een afzonderlijke kolom met een primaire sleutel, zoals "vriend-id" — een kunstmatige waarde die de toepassing gebruikt om de rijen van elkaar te onderscheiden.

Met behulp van een primaire sleutel kunt u relaties tussen meerdere tabellen definiëren. Stel dat de tabel "vrienden" een kolom "vriend-id" heeft die voor elke rij een uniek getal (de vriend-id) bevat. De gerelateerde tabel “telefoonnummers” kan in twee kolommen worden verdeeld, één met de “id_vriend” van de vriend die het desbetreffende telefoonnummer heeft, en één met het daadwerkelijke telefoonnummer. Op die manier kunnen alle telefoonnummers, ongeacht het aantal telefoonnummers per vriend, worden opgeslagen in de tabel “telefoonnummers” en via de primaire sleutel “id_vriend” worden gekoppeld aan de overeenkomstige vriend. Wanneer de primaire sleutel van een tabel wordt gebruikt in een gerelateerde tabel om de relatie tussen de records aan te geven, wordt de waarde in de gerelateerde tabel een externe sleutel genoemd. In tegenstelling tot vele databases biedt de lokale database-engine van AIR niet de mogelijkheid om beperkingen voor externe sleutels in te stellen. Deze beperkingen controleren automatisch of een ingevoegde of bijgewerkte externe-sleutelwaarde overeenkomt met een rij in de tabel met de primaire sleutel. Relaties op basis van externe sleutels vormen echter een belangrijk onderdeel van de structuur van een relationele database, en u wordt aangeraden externe sleutels te gebruiken wanneer u relaties tussen de tabellen van uw database definieert.

Informatie over SQL

Adobe AIR 1.0 of hoger

SQL (Structured Query Language) wordt bij relationele databases gebruikt om gegevens te bewerken en te zoeken. SQL is een beschrijvende en geen proceduretaal. Een SQL-instructie beschrijft de gegevensset die u zoekt maar geeft de computer geen instructies over hoe de gegevens moeten worden gezocht. De database-engine bepaalt hoe de gegevens worden gezocht.

De taal SQL is gestandaardiseerd door het ANSI (American National Standards Institute). De lokale SQL-database van Adobe AIR ondersteunt het grootste deel van de standaard SQL-92.

Zie “SQL-ondersteuning in lokale databases” op pagina 1176 voor specifieke beschrijvingen van de SQL-tall die wordt ondersteund in Adobe AIR.

Informatie over klassen van SQL-databases

Adobe AIR 1.0 of hoger

Als u met lokale SQL-databases wilt werken in ActionScript 3.0, moet u instanties van deze klassen uit het pakket flash.data gebruiken:

Klasse	Beschrijving
flash.data.SQLConnection	Biedt de mogelijkheid om databases (databasebestanden) te maken en te openen, evenals methoden voor het uitvoeren van bewerkingen op databaseniveau en het besturen van databasetransacties.
flash.data.SQLStatement	Vertegenwoordigt één SQL-instructie (één query of opdracht) die op een database wordt uitgevoerd, inclusief het definiëren van de instructietekst en het instellen van parameterwaarden.
flash.data.SQLResult	Biedt een manier om informatie over of het resultaat van de uitvoering van een instructie op te vragen, zoals de resulterende rijen na een `SELECT`-instructie, het aantal rijen dat wordt beïnvloed door een `UPDATE`- of `DELETE`-instructie, enzovoort.

Als u schema-informatie met een beschrijving van de structuur van een database wilt verkrijgen, moet u deze klassen uit het pakket flash.data gebruiken:

Klasse	Beschrijving
flash.data.SQLSchemaResult	Fungeert als container voor databaseschemaresultaten die worden gegenereerd door het aanroepen van de methode `SQLConnection.loadSchema()`.
flash.data.SQLTableSchema	Biedt informatie over één tabel in een database.
flash.data.SQLViewSchema	Biedt informatie over één weergave in een database.
flash.data.SQLIndexSchema	Biedt informatie over één kolom van een tabel of weergave in een database.
flash.data.SQLTriggerSchema	Biedt informatie over één trigger van een database.

Andere klassen uit het pakket flash.data bieden constanten die worden gebruikt met de klassen SQLConnection en SQLColumnSchema:

Klasse	Beschrijving
flash.data.SQLMode	Definieert een set van constanten voor de mogelijke waarden van de parameter `openMode` van de methoden `SQLConnection.open()` en `SQLConnection.openAsync()`.
flash.data.SQLColumnNameStyle	Definieert een set van constanten voor de mogelijke waarden van de eigenschap `SQLConnection.columnNameStyle`.
flash.data.SQLTransactionLockType	Definieert een set van constanten voor de mogelijke waarden van de optieparameter van de methode `SQLConnection.begin()`.
flash.data.SQLCollationType	Definieert een set van constanten voor de mogelijke waarden van de eigenschap `SQLColumnSchema.defaultCollationType` en de parameter `defaultCollationType` van de constructor `SQLColumnSchema()`.

Daarnaast vertegenwoordigen de volgende klassen uit het pakket flash.events de gebeurtenissen (en ondersteunende constanten) die u gebruikt:

Klasse	Beschrijving
flash.events.SQLEvent	Definieert de gebeurtenissen die door een SQLConnection- of SQLStatement-instantie worden verzonden wanneer een of meer bewerkingen uit de instantie met succes zijn uitgevoerd. Voor elke bewerking is een gekoppelde gebeurtenistypeconstante gedefinieerd in de klasse SQLEvent.
flash.events.SQLErrorEvent	Definieert de gebeurtenis die door een SQLConnection- of SQLStatement-instantie wordt verzonden wanneer een of meer bewerkingen uit de instantie een fout opleveren.
flash.events.SQLUpdateEvent	Definieert de gebeurtenis die door een SQLConnection-instantie wordt verzonden wanneer tabelgegevens in een van de gekoppelde databases veranderen door de uitvoering van de SQL-instructie `INSERT`, `UPDATE` of `DELETE`.

Ten slotte bieden de volgende klassen uit het pakket flash.errors informatie over bewerkingsfouten in de database:

Klasse	Beschrijving
flash.errors.SQLError	Biedt informatie over een bewerkingsfout in de database, zoals de bewerking die is mislukt en de oorzaak van de fout.
flash.errors.SQLErrorOperation	Definieert een set van constanten voor de mogelijke waarden van de eigenschap `operation` van de klasse SQLError, die de databasebewerking aangeeft waarbij de fout is opgetreden.

Informatie over synchrone en asynchrone uitvoeringsmodi

Adobe AIR 1.0 of hoger

Wanneer u code schrijft om met een lokale SQL-database te werken, selecteert u een van de twee beschikbare uitvoeringsmodi voor bewerkingen in de database: synchroon of asynchroon. De codevoorbeelden tonen op algemene manier hoe u elke bewerking op beide manieren kunt uitvoeren zodat u het voorbeeld kunt gebruiken dat het meest geschikt is voor uw behoeften.

In asynchrone uitvoeringsmodus geeft u de runtime een instructie, waarna de runtime een gebeurtenis verzendt wanneer de desbetreffende bewerking is voltooid of mislukt. Eerst geeft u de database-engine de opdracht een bewerking uit te voeren. De database-engine werkt op de achtergrond terwijl de toepassing verderwerkt. Wanneer ten slotte de bewerking is voltooid (of mislukt), verzendt de database-engine een gebeurtenis. Uw code, geactiveerd door de gebeurtenis, voert indien nodig daarop volgende bewerkingen uit. Deze procedure heeft een groot voordeel: de runtime voert de databasebewerkingen op de achtergrond uit terwijl de hoofdtoepassingscode verder wordt uitgevoerd. Als de databasebewerking lange tijd duurt, blijft de toepassing verderwerken. En het belangrijkste is dat interactie met de gebruiker mogelijk blijft omdat het scherm niet bevriest. Het nadeel is dat asynchrone bewerkingscode soms complexer is om te schrijven. Dit is doorgaans het geval wanneer meerdere afhankelijke bewerkingen moeten worden verdeeld tussen verschillende gebeurtenislistenermethoden.

Conceptueel is het eenvoudiger om bewerkingen als één sequentie van stappen (een set van synchrone bewerkingen) te beschrijven in plaats van als een set van bewerkingen verdeeld in meerdere gebeurtenislistenermethoden. Naast asynchrone databasebewerkingen is met Adobe AIR ook het synchroon uitvoeren van databasebewerkingen mogelijk. In de synchrone uitvoeringsmodus worden de bewerkingen niet op de achtergrond uitgevoerd. In plaats daarvan worden ze uitgevoerd volgens dezelfde uitvoeringssequentie als alle andere toepassingscode. U geeft de database-engine de opdracht een bewerking uit te voeren. Vervolgens wordt de code op dat punt onderbroken terwijl de database-engine zijn werk doet. Nadat de bewerking is voltooid, gaat de uitvoering verder vanaf de volgende regel van uw code.

U bepaalt op SQLConnection-niveau welke uitvoeringsmodus (synchroon of asynchroon) voor bewerkingen wordt gebruikt. Het is niet mogelijk om via één databaseverbinding bepaalde bewerkingen of instructies synchroon en andere asynchroon uit te voeren. U bepaalt welke uitvoeringsmodus (synchroon of asynchroon) voor een SQLConnection wordt gebruikt door een SQLConnection-methode op te roepen om de database te openen. Als u `SQLConnection.open()` oproept, werkt de verbinding in synchrone uitvoeringsmodus. Roept u `SQLConnection.openAsync()` op, dan werkt de verbinding in asynchrone uitvoeringsmodus. Nadat een SQLConnection-instantie met een database is verbonden met behulp van `open()` of `openAsync()`, wordt deze in synchrone of asynchrone uitvoeringsmodus vergrendeld, tenzij u de verbinding met de database sluit en weer opent.

Elke uitvoeringsmodus heeft specifieke voordelen. De meeste aspecten van beide modi zijn identiek. Er zijn echter bepaalde verschillen waarmee u het best rekening houdt terwijl u in een bepaalde modus werkt. Zie "Synchrone en asynchrone databasebewerkingen gebruiken" op pagina 795 voor meer informatie over deze onderwerpen en suggesties voor het werken in elke modus.

Databases creëren en wijzigen

Adobe AIR 1.0 of hoger

Voordat uw toepassing gegevens kan toevoegen of ophalen, moet er een database met gedefinieerde tabellen aanwezig zijn en moet de database toegankelijk zijn voor uw toepassing. Hieronder wordt beschreven hoe u een database creëert en de gegevensstructuur binnen een database definieert. Deze taken zijn weliswaar minder vaak nodig dan het invoegen en ophalen van gegevens maar zijn toch noodzakelijk voor de meeste toepassingen.

Meer Help-onderwerpen

Mind the Flex: een bestaande AIR-database bijwerken

Database creëren

Adobe AIR 1.0 of hoger

Voordat u een databasebestand kunt maken, moet u eerst een SQLConnection-instantie maken. Roep de methode `open()` op om de instantie in synchrone uitvoeringsmodus te openen, of roep de methode `openAsync()` op om de instantie in asynchrone uitvoeringsmodus te openen. De methoden `open()` en `openAsync()` worden gebruikt om een verbinding met een database te openen. Als u een File-instantie gebruikt die verwijst naar een niet-bestaande bestandslocatie voor de parameter `reference` (de eerste parameter), creëert de methode `open()` of `openAsync()` een databasebestand op die bestandslocatie en wordt een verbinding met de zojuist gecreëerde database geopend.

Ongeacht of u de methode `open()` of de methode `openAsync()` aanroept om een database te maken, de naam van het databasebestand kan een willekeurige bestandsnaam zijn met een willekeurige bestandsnaamextensie. Als u de methode `open()` of `openAsync()` oproept met de instelling `null` voor de parameter`reference`, wordt een nieuwe database in het geheugen gecreëerd in plaats van een databasebestand op de vaste schijf.

De volgende code illustreert het creëren van een databasebestand (een nieuwe database) via de asynchrone uitvoeringsmodus. In dit geval wordt het databasebestand opgeslagen in de "De opslagmap van een toepassing aanwijzen" op pagina 710 en krijgt het bestand de naam "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>
```

Opmerking: *Hoewel u met de File-klasse naar een native bestandspad kunt wijzen, kan dit leiden naar toepassingen die niet op alle platformen werken. Het pad C:\Documents and Settings\joe\test.db werkt bijvoorbeeld alleen op Windows. Hierdoor kunt u het best de statische eigenschappen van de klasse File gebruiken, zoals* `File.applicationStorageDirectory` *en de methode* `resolvePath()` *(zoals afgebeeld in het vorige voorbeeld). Zie* *"Paden van File-objecten" op pagina 706* *voor meer informatie.*

Als u bewerkingen synchroon wilt uitvoeren, roept u de methode `open()` op wanneer u een databaseverbinding opent met de SQLConnection-instantie. In het volgende voorbeeld ziet u hoe u een SQLConnection-instantie creëert en opent die de bewerkingen synchroon uitvoert:

```
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>
```

Databasetabellen creëren

Adobe AIR 1.0 of hoger

Als u een tabel in een database wilt maken, moet u een SQL-instructie uitvoeren op de desbetreffende database via dezelfde procedure die u gebruikt voor het uitvoeren van een SQL-instructie zoals `SELECT`, `INSERT` enzovoort. U creëert een tabel met behulp van de instructie `CREATE TABLE`, waarbij u kolomdefinities en beperkingen voor de nieuwe tabel opgeeft. Zie "Werken met SQL-instructies" op pagina 770 voor meer informatie over het uitvoeren van SQL-instructies.

In het volgende voorbeeld ziet u hoe u een tabel met de naam "employees" in een bestaand databasebestand creëert via de asynchrone uitvoeringsmodus. Let op: deze code gaat ervan uit dat een SQLConnection-instantie met de naam `conn` al is gemaakt en met een database is verbonden.

```
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>
```

In het volgende voorbeeld ziet u hoe u een tabel met de naam “employees” in een bestaand databasebestand maakt via de synchrone uitvoeringsmodus. Let op: deze code gaat ervan uit dat een SQLConnection-instantie met de naam `conn` al is gemaakt en met een database is verbonden.

```
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>
```

Werken met SQL-databasegegevens

Adobe AIR 1.0 of hoger

Bij het werken met lokale SQL-databases voert u regelmatig dezelfde taken uit, Tot deze taken behoren het tot stand brengen van een verbinding met een database, het toevoegen van gegevens aan tabellen, en het ophalen van gegevens uit tabellen in een database. Bij het uitvoeren van deze taken moet u een aantal aspecten in acht nemen, zoals het werken met gegevenstypen en het verhelpen van fouten.

Daarnaast zijn er verschillende databasetaken die u niet zo vaak zult uitvoeren maar doorgaans wel voordat u de veel voorkomende taken kunt uitvoeren. U moet bijvoorbeeld een database creëren en de structuur van de tabellen in de database definiëren voordat u kunt verbinden met de database en gegevens uit tabellen kunt ophalen. Deze minder frequente, initiële set-uptaken worden besproken in “Databases creëren en wijzigen” op pagina 759.

U kunt kiezen om databasebewerkingen asynchroon uit te voeren, wat betekent dat de database-engine op de achtergrond wordt uitgevoerd en deze u meldt wanneer de bewerking is voltooid (of mislukt) door een gebeurtenis te verzenden. U kunt deze bewerkingen echter ook synchroon uitvoeren. In dat geval worden de databasebewerkingen na elkaar uitgevoerd en wacht de hele toepassing (ook het vernieuwen van het scherm) tot de bewerkingen zijn voltooid voordat andere code wordt uitgevoerd. Zie "Synchrone en asynchrone databasebewerkingen gebruiken" op pagina 795 voor meer informatie over het werken in synchrone of asynchrone uitvoeringsmodus.

Verbinden met een database

Adobe AIR 1.0 of hoger

Voordat u databasebewerkingen kunt uitvoeren, moet u een verbinding met het databasebestand tot stand brengen. Er wordt een SQLConnection-instantie gebruikt die een verbinding met een of meer databases vertegenwoordigt. De eerste database waarmee u via een SQLConnection-instantie verbinding maakt, wordt de "hoofd"database genoemd. Met deze database maakt u verbinding via de methode `open()` (voor de synchrone uitvoeringsmodus) of `openAsync()` (voor de asynchrone uitvoeringsmodus).

Als u een database opent via de bewerking `openAsync()`, moet u de gebeurtenis `open` van de SQLConnection-instantie registreren zodat u een melding ontvangt wanneer de bewerking `openAsync()` is voltooid. Deze registratie is nodig als u wilt dat de gebeurtenis `error` van de SQLConnection-instantie controleert of de bewerking is voltooid/mislukt.

In het volgende voorbeeld ziet u hoe u een bestaand databasebestand opent voor asynchrone uitvoering. Het databasebestand heeft de naam "DBSample.db" en bevindt zich in de "De opslagmap van een toepassing aanwijzen" op pagina 710.

```
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>
```

In het volgende voorbeeld ziet u hoe u een bestaand databasebestand opent voor synchrone uitvoering. Het databasebestand heeft de naam "DBSample.db" en bevindt zich in de “De opslagmap van een toepassing aanwijzen” op pagina 710 van de gebruiker.

```
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>
```

Let op: bij het oproepen van de methode `openAsync()` in het asynchrone voorbeeld en het oproepen van de methode `open()` in het synchrone voorbeeld is het tweede argument de constante `SQLMode.UPDATE`. Als u `SQLMode.UPDATE` opgeeft voor de tweede parameter (`openMode`), treedt een fout op als het opgegeven bestand niet bestaat. Als u `SQLMode.CREATE` opgeeft voor de parameter `openMode` (of als u geen waarde opgeeft voor de parameter `openMode`), probeert de runtime een databasebestand te creëren als het opgegeven bestand niet bestaat. Als het bestand echter bestaat, wordt het geopend (vergelijkbaar met het gebruik van `SQLMode.Update`). U kunt ook `SQLMode.READ` opgeven voor de parameter `openMode` als u een bestaande database in alleen-lezen modus wilt openen. In dat geval kunnen gegevens uit de database worden opgehaald maar niet worden toegevoegd, verwijderd of gewijzigd.

Werken met SQL-instructies

Adobe AIR 1.0 of hoger

Een individuele SQL-instructie (een query of opdracht) wordt in de runtime vertegenwoordigd door een SQLStatement-object. Ga als volgt te werk om een SQL-instructie te creëren en uit te voeren:

Creëer een SQLStatement-instantie.
Het SQLStatement-object vertegenwoordigt de SQL-instructie in uw toepassing.

```
var selectData:SQLStatement = new SQLStatement();
```

Geef de database op waarop u de query wilt toepassen.
Hiervoor stelt u de eigenschap `sqlConnection` van het SQLStatement-object in op de SQLConnection-instantie die met de gewenste database is verbonden.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

Geef de daadwerkelijke SQL-instructie op.
Creëer de instructietekst als een tekenreeks (String) en wijs deze toe aan de eigenschap `text` van de SQLStatement-instantie.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

Definieer functies om het resultaat van de uitvoeringsbewerking (alleen asynchrone uitvoeringsmodus) te verwerken.
Gebruik de methode `addEventListener()` om functies als listeners voor de gebeurtenissen `result` en `error` van de SQLStatement-instantie te registreren.

```
// 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
}
```

U kunt ook listenermethoden opgeven met behulp van een Responder-object. In dat geval creëert u de Responder-instantie en koppelt u de listenermethoden aan die instantie.

```
// 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
}
```

Als de instructietekst parameterdefinities bevat, wijst u waarden toe aan de desbetreffende parameters.
U wijst parameterwaarden toe via de associatieve-array-eigenschap `parameters` van de SQLStatement-instantie.

```
selectData.parameters[":param1"] = 25;
```

Voer de SQL-instructie uit.
Roep de methode `execute()` van de SQLStatement-instantie op.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

Als u een Responder in plaats van gebeurtenislisteners gebruikt in asynchrone uitvoeringsmodus, moet u ook de Responder-instantie opgeven voor de methode `execute()`.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

Zie de volgende onderwerpen voor specifieke voorbeelden die deze procedurestappen illustreren:

"Gegevens ophalen uit een database" op pagina 774

"Gegevens invoegen" op pagina 784

"Gegevens wijzigen of verwijderen" op pagina 790

Parameters gebruiken in instructies

Adobe AIR 1.0 of hoger

Met SQL-instructieparameters kunt u een herbruikbare SQL-instructie creëren. Bij het gebruik van instructieparameters kunnen waarden binnen de instructie wijzigen (zoals waarden die worden toegevoegd aan een instructie van het type `INSERT`) maar blijft de basistekst van de instructie ongewijzigd. Dat betekent dat het gebruik van parameters prestatievoordelen biedt en het coderen van een toepassing gemakkelijker maakt.

Instructieparameters

Adobe AIR 1.0 of hoger

Een toepassing gebruikt vaak dezelfde SQL-instructie meerdere keren, met slechts een kleine wijziging. Bijvoorbeeld: een voorraadbeheertoepassing waarmee een gebruiker nieuwe voorraaditems kan toevoegen aan de database. De toepassingscode die een voorraaditem toevoegt aan de database voert de SQL-instructie `INSERT` uit, waarmee de gegevens daadwerkelijk worden toegevoegd aan de database. Elke keer dat de instructie wordt uitgevoerd, is er echter een kleine wijziging. Met name de daadwerkelijke waarden die worden ingevoegd in de tabel, verschillen omdat ze specifiek zijn voor het voorraaditem dat wordt toegevoegd.

Als een SQL-instructie meerdere keren maar met verschillende waarden wordt gebruikt, wordt u aangeraden een SQL-instructie te gebruiken die met parameters werkt en niet met literale waarden in de SQL-tekst. Een parameter is een placeholder in de instructietekst die door een daadwerkelijke waarde wordt vervangen elke keer dat de instructie wordt uitgevoerd. Als u parameters in een SQL-instructie wilt gebruiken, maakt u de SQLStatement-instantie op de gewone manier. Voor de daadwerkelijke SQL-instructie die aan de eigenschap `text` wordt toegewezen, gebruikt u parameterplaceholders in plaats van literale waarden. Vervolgens definieert u de waarde voor elke parameter door de waarde van een element in de eigenschap `parameters` van de SQLStatement-instantie in te stellen. Aangezien de eigenschap `parameters` een associatieve array is, stelt u een specifieke waarde in met de volgende syntaxis:

```
statement.parameters[parameter_identifier] = value;
```

parameter_identifier is een tekenreeks als u een benoemde parameter gebruikt, of een index van gehele getallen als u een onbenoemde parameter gebruikt.

Benoemde parameters gebruiken

Adobe AIR 1.0 of hoger

Een parameter kan benoemd zijn. Een benoemde parameter heeft een specifieke naam die door de database wordt gebruikt om de parameterwaarde te koppelen aan de overeenkomstige placeholderlocatie in de instructietekst. De naam van een parameter bestaat uit het teken ":" of "@" gevolgd door een naam, zoals in de volgende voorbeelden wordt geïllustreerd:

```
:itemName
@firstName
```

De volgende code illustreert het gebruik van benoemde parameters:

```
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();
```

Onbenoemde parameters gebruiken

Adobe AIR 1.0 of hoger

Als alternatief voor het gebruik van benoemde parameters kunt u ook onbenoemde parameters gebruiken. Als u een onbenoemde parameter wilt gebruiken, geeft u een parameter in een SQL-instructie aan met een "?" (vraagteken). Aan elke parameter wordt een numerieke index toegewezen, afhankelijk van de volgorde van de parameters in de instructie, te beginnen bij index 0 voor de eerste parameter. In het volgende voorbeeld ziet u een variatie van het vorige voorbeeld, nu met onbenoemde parameters:

```
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();
```

Voordelen van het gebruik van parameters

Adobe AIR 1.0 of hoger

Het gebruik van parameters in een SQL-instructie biedt verschillende voordelen:

Betere prestaties Een SQLStatement-instantie die parameters gebruikt, kan efficiënter werken dan een instantie die de SQL-tekst dynamisch genereert bij elke uitvoering. De prestatieverbetering is te danken aan het feit dat de instructie één keer wordt voorbereid en vervolgens meerdere keren met verschillende parameterwaarden kan worden uitgevoerd zonder dat de SQL-instructie opnieuw hoeft te worden gecompileerd.

Type toewijzen aan expliciete gegevens Parameters worden gebruikt voor het omzetten van waarden in een toegewezen type als de daadwerkelijke waarden niet bekend zijn op het moment dat de SQL-instructie wordt geschreven. Het gebruik van parameters is de enige manier om de opslagklasse te garanderen van een waarde die in de database wordt opgenomen. Als geen parameters worden gebruikt, probeert de runtime alle waarden om te zetten van hun tekstvorm in een opslagklasse op basis van het type van de overeenkomstige kolom.

Zie "Ondersteuning van gegevenstypen" op pagina 1199 voor meer informatie over opslagklassen en kolomaffiniteit.

Betere beveiliging Het gebruik van parameters helpt een schadelijke techniek die SQL-injectieaanval wordt genoemd, onmogelijk te maken. Bij een SQL-injectieaanval voert de gebruiker SQL-code in via een voor de gebruiker toegankelijke locatie (zoals een gegevensinvoerveld). Als toepassingscode een SQL-instructie genereert door gebruikersinvoer rechtstreeks samen te voegen tot SQL-tekst, wordt de door de gebruiker ingevoerde SQL-code uitgevoerd op de database. De volgende code is een voorbeeld van het samenvoegen van gebruikersinvoer tot SQL-tekst. **Gebruik deze techniek niet**:

```
// 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;
```

Door instructieparameters in plaats van door de gebruiker ingevoerde waarden samen te voegen tot de tekst voor een instructie, maakt u een SQL-injectieaanval onmogelijk. Er kan geen SQL-injectie plaatsvinden omdat de parameterwaarden expliciet als vervangen waarden worden verwerkt, en niet als waarden die een onderdeel van de literale instructietekst worden. Dit is het aanbevolen alternatief voor de bovenstaande code:

```
// 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;
```

Gegevens ophalen uit een database

Adobe AIR 1.0 of hoger

Het ophalen van gegevens uit een database bestaat uit twee stappen. Eerst voert u de SQL-instructie `SELECT` uit om de gegevensset te beschrijven die u uit de database wilt ophalen. Vervolgens benadert u de opgehaalde gegevens, en geeft u ze weer of bewerkt u ze, afhankelijk van de toepassing.

SELECT-instructies uitvoeren

Adobe AIR 1.0 of hoger

Voor het ophalen van bestaande gegevens uit een database gebruikt u een SQLStatement-instantie. Wijs de juiste SQL-instructie `SELECT` toe aan de eigenschap `text` van de instantie en roep vervolgens de methode `execute()` aan.

Voor gedetailleerde informatie over de syntaxis van de instructie `SELECT` gaat u naar "SQL-ondersteuning in lokale databases" op pagina 1176.

Het volgende voorbeeld demonstreert de uitvoering van een `SELECT`-instructie om gegevens op te halen uit een tabel met de naam 'products' met de asynchrone uitvoeringsmodus:

```
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>
```

Het volgende voorbeeld demonstreert de uitvoering van een `SELECT`-instructie om gegevens op te halen uit een tabel met de naam 'products' met de synchrone uitvoeringsmodus:

```
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>
```

Nadat in de asynchrone uitvoeringsmodus de uitvoering van de instructie is voltooid, verzendt de SQLStatement-instantie de gebeurtenis `result` (`SQLEvent.RESULT`) om aan te geven dat de instructie correct is uitgevoerd. Als een Responder-object echter als argument wordt doorgegeven aan de methode `execute()`, wordt de resultaathandlerfunctie van het Responder-object aangeroepen. In de synchrone uitvoeringsmodus wordt de uitvoering onderbroken tot de bewerking `execute()` is voltooid. Daarna gaat de uitvoering verder vanaf de volgende coderegel.

Resultaatgegevens van SELECT-instructie benaderen

Adobe AIR 1.0 of hoger

Nadat de uitvoering van de instructie `SELECT` is voltooid, kunnen de opgehaalde gegevens worden benaderd. Na het uitvoeren van een `SELECT`-instructie kunt u de resulterende gegevens ophalen door de methode `getResult()` van het SQLStatement-object aan te roepen:

```
var result:SQLResult = selectStatement.getResult();
```

De methode `getResult()` retourneert een SQLResult-object. De eigenschap `data` van het SQLResult-object is een array die de resultaten van de instructie `SELECT` bevat:

```
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];
}
```

Elke gegevensrij in de resultaatset van `SELECT` wordt een Object-instantie die is opgenomen in de array `data`. Het desbetreffende object heeft eigenschappen waarvan de naam overeenkomt met de naam van de kolommen in de resultaatset. De eigenschappen bevatten de waarden uit de kolommen in de resultaatset. De instructie `SELECT` geeft bijvoorbeeld een resultaatset met drie kolommen op: “itemId”, “itemName” en “price”. Voor elke rij van de resultaatset wordt een Object-instantie gecreëerd met de eigenschapnamen `itemId`, `itemName` en `price`. Deze eigenschappen bevatten de waarden uit de overeenkomstige kolommen.

De volgende code definieert een SQLStatement-instantie waarvan de tekst een `SELECT`-instructie is. De instructie haalt rijen op die de waarden uit de kolommen `firstNamelastName` bevatten op alle rijen uit de tabel `employees`. Dit voorbeeld gebruikt de asynchrone uitvoeringsmodus. Nadat de uitvoering is voltooid, wordt de methode `selectResult()` opgeroepen en worden de resultaatrijen met gegevens benaderd met `SQLStatement.getResult()` en weergegeven via de methode `trace()`. Let op: deze code gaat ervan uit dat een SQLConnection-instantie met de naam `conn` al is gedefinieerd en met de database is verbonden. De code gaat er ook van uit dat de tabel "employees" al is gecreëerd en gegevens bevat.

```
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>
```

De volgende code illustreert dezelfde technieken als de vorige code maar in de synchrone uitvoeringsmodus. Het voorbeeld definieert een SQLStatement-instantie waarvan de tekst een `SELECT`-instructie is. De instructie haalt rijen op met de kolomwaarden `firstName` en `lastName` van alle rijen uit de tabel `employees`. De resultaatrijen met gegevens worden benaderd met `SQLStatement.getResult()` en weergegeven via de methode `trace()`. Let op: deze code gaat ervan uit dat een SQLConnection-instantie met de naam `conn` al is gedefinieerd en met de database is verbonden. De code gaat er ook van uit dat de tabel "employees" al is gecreëerd en gegevens bevat.

```
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>
```

Het gegevenstype van SELECT-resultaatgegevens definiëren

Adobe AIR 1.0 of hoger

Elke rij die het resultaat is van een `SELECT`-instructie, wordt standaard gegenereerd als een Object-instantie met eigenschappen die zijn benoemd volgens de naam van de kolommen in de resultaatset en met de waarde van elke kolom als de waarde van de overeenkomstige eigenschap. Voordat u echter de SQL-instructie `SELECT` uitvoert, kunt u de eigenschap `itemClass` van de SQLStatement-instantie instellen op een klasse. Als u de eigenschap `itemClass` instelt, wordt elke rij die het resultaat is van de instructie `SELECT` gegenereerd als een instantie van de opgegeven klasse. De runtime wijst de waarden van resultaatkolommen aan eigenschapwaarden toe door de naam van de kolommen in de resultaatset van `SELECT` te baseren op de naam van de eigenschappen in de klasse `itemClass`.

Elke klasse die is toegewezen als een `itemClass`-eigenschapwaarde, moet een constructor hebben waarvoor geen parameters nodig zijn. Bovendien moet de klasse één eigenschap hebben voor elke kolom die het resultaat is van de instructie `SELECT`. Als een kolom in de lijst `SELECT` geen gekoppelde eigenschapnaam in de klasse `itemClass` heeft, wordt dit als een fout beschouwd.

SELECT-resultaten in delen ophalen

Adobe AIR 1.0 of hoger

Bij de uitvoering van een `SELECT`-instructie worden standaard alle rijen van de resultaatset tegelijk opgehaald. Nadat de instructie is voltooid, verwerkt u doorgaans de opgehaalde gegevens, bijvoorbeeld door objecten te creëren of de gegevens op het scherm weer te geven. Als de instructie een groot aantal rijen genereert, kan het verwerken van alle gegevens tegelijk een grote belasting voor de computer zijn, waardoor de gebruikersinterface niet wordt vernieuwd.

U kunt de zichtbare prestaties van uw toepassing verbeteren door te zorgen dat de runtime slechts een bepaald aantal resultaatrijen tegelijk weergeeft. Hierdoor worden de initiële resultaatgegevens sneller weergegeven. Op deze manier kunt u ook de resultaatrijen in sets verdelen zodat de gebruikersinterface wordt bijgewerkt telkens wanneer een set rijen is verwerkt. Let op: deze techniek is alleen nuttig in asynchrone uitvoeringsmodus.

Als u de resultaten van `SELECT` in delen wilt ophalen, geeft u een waarde op voor de eerste parameter van de methode `SQLStatement.execute()` (de parameter `prefetch`). De parameter `prefetch` geeft het aantal rijen aan dat u wilt ophalen de eerste keer dat de instructie wordt uitgevoerd. Wanneer u de methode `execute()` van een SQLStatement-instantie aanroept, geeft u een `prefetch`-parameterwaarde op, waarna alleen dat aantal rijen wordt opgehaald:

```
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
```

De instructie verzendt de gebeurtenis `result` wanneer de eerste set resultaatrijen beschikbaar is. De resulterende eigenschap `data` van de SQLResult-instantie bevat de gegevensrijen en de eigenschap `complete` ervan geeft aan of er nog op te halen resultaatrijen resteren. Als u de volgende set resultaatrijen wilt ophalen, roept u de methode `next()` van de SQLStatement-instantie op. Net als bij de methode `execute()` wordt de eerste parameter van de methode `next()` gebruikt om aan te geven hoeveel rijen moeten worden opgehaald de volgende keer dat de gebeurtenis result wordt verzonden.

```
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);
        }
    }
}
```

De SQLStatement verzendt de gebeurtenis `result` elke keer dat de methode `next()` een volgende set resultaatrijen weergeeft. Dit betekent dat dezelfde listenerfunctie kan worden gebruikt om verder te gaan met het verwerken van resultaten (op basis van `next()`-oproepen) tot alle rijen zijn opgehaald.

Zie de beschrijvingen voor de methode `SQLStatement.execute()` (de `prefetch`-parameterbeschrijving) en de methode `SQLStatement.next()` voor meer informatie

Gegevens invoegen

Adobe AIR 1.0 of hoger

Voor het toevoegen van gegevens aan een database moet de SQL-instructie `INSERT` worden uitgevoerd. Nadat de instructie is uitgevoerd, hebt u toegang tot de primaire sleutel voor de zojuist ingevoegde rij als de sleutel door de database is gegenereerd.

INSERT-instructies uitvoeren

Adobe AIR 1.0 of hoger

Als u gegevens wilt toevoegen aan een tabel in een database, moet u een SQLStatement-instantie maken en uitvoeren waarvan de tekst de SQL-instructie `INSERT` is.

In het volgende voorbeeld ziet u hoe u een SQLStatement-instantie gebruikt om een gegevensrij toe te voegen aan de bestaande tabel "employees". Dit voorbeeld illustreert het invoegen van gegevens in asynchrone uitvoeringsmodus. Let op: deze vermelding ervan uit dat een SQLConnection-instantie met de naam `conn` al is gemaakt en met een database is verbonden. De code gaat er ook van uit dat de tabel "employees" al is gecreëerd.

```
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>
```

In het volgende voorbeeld ziet u hoe u een gegevensrij toevoegt aan de bestaande tabel “employees” in synchrone uitvoeringsmodus. Let op: deze vermelding ervan uit dat een SQLConnection-instantie met de naam `conn` al is gemaakt en met een database is verbonden. De code gaat er ook van uit dat de tabel “employees” al is gecreëerd.

```
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>
```

De door de database gegenereerde primaire sleutel van een ingevoegde rij ophalen

Adobe AIR 1.0 of hoger

Vaak moet uw code na het invoegen van een gegevensrij in een tabel de door de database gegenereerde primaire sleutel of de rij-identificatiewaarde voor de zojuist ingevoegde rij weten. Nadat u een rij in één tabel hebt ingevoegd, wilt u bijvoorbeeld rijen toevoegen aan een gerelateerde tabel. In dat geval kunt u het best de waarde van de primaire sleutel als vreemde sleutel in de gerelateerde tabel invoegen. De primaire sleutel van de zojuist ingevoegde rij kan worden opgehaald via het SQLResult-object dat gekoppeld is aan het uitvoeren van de instructie. Dit object wordt ook gebruikt om resultaatgegevens te benaderen nadat de instructie `SELECT` is uitgevoerd. Net als bij alle andere SQL-instructies creëert de runtime na het voltooien van de instructie `INSERT` een SQLResult-instantie. U benadert de SQLResult-instantie door de methode `getResult()` van het SQLStatement-object aan te roepen als u een gebeurtenislistener of de synchrone uitvoeringsmodus gebruikt. Als u echter de asynchrone uitvoeringsmodus gebruikt en u een Responder-instantie doorgeeft aan de aanroep `execute()`, wordt de SQLResult-instantie als argument doorgegeven aan de resultaathandlerfunctie. In elk geval heeft de SQLResult-instantie de eigenschap `lastInsertRowID`, die de rij-identificator van de laatst ingevoegde rij bevat als de uitgevoerde SQL-instructie de instructie `INSERT` is.

In het volgende voorbeeld ziet u hoe u de primaire sleutel van een ingevoegde rij benadert in asynchrone uitvoeringsmodus:

```
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
}
```

In het volgende voorbeeld ziet u hoe u de primaire sleutel van een ingevoegde rij benadert in synchrone uitvoeringsmodus:

```
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
}
```

Let op: mogelijk heeft de rij-identificator de waarde van de kolom die in de tabeldefinitie is opgegeven als de kolom met de primaire sleutel. Hierbij gelden de volgende regels:

- Als voor de tabel een kolom met de primaire sleutel is gedefinieerd waarbij de kolomgegevens van het type `INTEGER` zijn, bevat de eigenschap `lastInsertRowID` de waarde die in de desbetreffende rij is ingevoegd (of de waarde die door de runtime is gegenereerd als het een kolom van het type `AUTOINCREMENT` betreft).
- Als voor de tabel meerdere kolommen met een primaire sleutel zijn gedefinieerd (een samengestelde sleutel) of als voor de tabel één kolom is gedefinieerd met een primaire sleutel waarbij de kolomgegevens niet van het type `INTEGER` zijn, genereert de database een onzichtbare integer als rij-identificatiewaarde voor de rij. Die gegenereerde waarde is de waarde van de eigenschap `lastInsertRowID`.
- De waarde is altijd de rij-identificator van de laatst ingevoegde rij. Als de instructie `INSERT` een trigger activeert waardoor vervolgens een rij wordt ingevoegd, bevat de eigenschap `lastInsertRowID` de rij-identificator van de rij die het laatst door de trigger is ingevoegd en niet van de rij die door de instructie `INSERT` is gemaakt.

Dit betekent dat als u een kolom met een expliciet gedefinieerde primaire sleutel wilt waarvan de waarde beschikbaar is na de opdracht `INSERT` via de eigenschap `SQLResult.lastInsertRowID`, u de kolom moet definiëren als een kolom van het type `INTEGER PRIMARY KEY`. Zelfs als uw tabel geen expliciete kolom van het type `INTEGER PRIMARY KEY` bevat, is het geen probleem om de door de database gegenereerde rij-identificator als primaire sleutel voor uw tabel te gebruiken voor het definiëren van relaties met gerelateerde tabellen. De waarde van de kolom met de rij-identificator is in alle SQL-instructies beschikbaar door het gebruik van een van de speciale kolomnamen: `ROWID`, `_ROWID_` of `OID`. U kunt in een gerelateerde tabel een kolom met een vreemde sleutel creëren en de rij-identificatiewaarde gebruiken als waarde voor de kolom met de vreemde sleutel, zoals bij een expliciet gedefinieerde kolom van het type `INTEGER PRIMARY KEY`. Als u dus een willekeurige primaire sleutel gebruikt in plaats van een natuurlijke sleutel en u het niet erg vindt dat de runtime de waarde van de primaire sleutel voor u genereert, is er geen groot verschil tussen het gebruik van een kolom van het type `INTEGER PRIMARY KEY` of van de door het systeem gegenereerde rij-identificator als primaire sleutel van de tabel bij het definiëren van een relatie met vreemde sleutel tussen twee tabellen.

Zie "SQL-ondersteuning in lokale databases" op pagina 1176 voor meer informatie over primaire sleutels en gegenereerde rij-id's.

Gegevens wijzigen of verwijderen

Adobe AIR 1.0 of hoger

Voor het uitvoeren van andere gegevensbewerkingen gebruikt u dezelfde procedure als voor het uitvoeren van de SQL-instructie `SELECT` of `INSERT`. Een beschrijving van deze procedure vindt u in "Werken met SQL-instructies" op pagina 770. Vervang gewoon de SQL-instructie in de eigenschap `text` van de SQLStatement-instantie:

- Als u bestaande gegevens in een tabel wilt wijzigen, gebruikt u de instructie `UPDATE`.
- Als u een of meer gegevensrijen uit een tabel wilt verwijderen, gebruikt u de instructie `DELETE`.

Zie "SQL-ondersteuning in lokale databases" op pagina 1176 voor beschrijvingen van deze instructies.

Werken met meerdere databases

Adobe AIR 1.0 of hoger

Gebruik de methode `SQLConnection.attach()` om een verbinding met een andere database te openen voor een SQLConnection-instantie die al een open database heeft. Geef de verbonden database een naam met behulp van de parameter "name" via het oproepen van de methode `attach()`. Wanneer u vervolgens instructies voor het bewerken van de desbetreffende database schrijft, kunt u die naam in een voorvoegsel gebruiken (met de syntaxis `databasenaam.tabelnaam`) om eventuele tabelnamen in uw SQL-instructies te definiëren, zodat de runtime weet dat de tabel zich in de opgegeven database bevindt.

U kunt werken met één SQL-instructie die verwijst naar tabellen uit meerdere databases die zijn verbonden met dezelfde SQLConnection-instantie. Als een transactie voor de SQLConnection-instantie wordt gecreëerd, geldt de desbetreffende transactie voor alle SQL-instructies die met behulp van de SQLConnection-instantie worden uitgevoerd. Deze regel geldt altijd, ongeacht de database waarop de instructie wordt uitgevoerd.

U kunt echter ook meerdere SQLConnection-instanties in een toepassing creëren, waarbij elke instantie is verbonden met een of meer databases. Als u echter meerdere verbindingen met dezelfde database gebruikt, mag u niet vergeten dat databasetransacties niet worden gedeeld over SQLConnection-instanties. Dit betekent dat als u via meerdere SQLConnection-instanties verbinding maakt met hetzelfde databasebestand, u er niet op mag vertrouwen dat de wijzigingen in de gegevens van beide verbindingen op de verwachte manier worden doorgevoerd. Als bijvoorbeeld twee `UPDATE`- of `DELETE`-instructies op dezelfde database maar via verschillende SQLConnection-instanties worden uitgevoerd en er een toepassingsfout optreedt na één bewerking, bevinden de databasegegevens zich mogelijk in een onomkeerbare tussenfase, waardoor de integriteit van de database (en dus ook van de toepassing) in gevaar kan komen.

Databasefouten verhelpen

Adobe AIR 1.0 of hoger

Over het algemeen verloopt het verhelpen van databasefouten zoals het verhelpen van andere runtimefouten. U wordt aangeraden code te schrijven die is voorbereid op het optreden van mogelijke fouten en deze kan verhelpen, in plaats van dit door de runtime te laten doen. Databasefouten worden doorgaans in drie categorieën verdeeld: verbindingsfouten, SQL-syntaxisfouten en beperkingsfouten.

Verbindingsfouten

Adobe AIR 1.0 of hoger

De meeste databasefouten zijn verbindingsfouten. Deze kunnen tijdens alle bewerkingen optreden. Er zijn weliswaar strategieën voor het voorkomen van verbindingsfouten maar er is slechts zelden een eenvoudige manier om zonder problemen van een verbindingsfout te herstellen als de database een belangrijk deel is van uw toepassing.

De meeste verbindingsfouten worden veroorzaakt door de manier waarop de runtime interageert met het besturingssysteem, het bestandssysteem en het databasebestand. Er treedt bijvoorbeeld een verbindingsfout op als de gebruiker geen toestemming heeft om een databasebestand te creëren op een bepaalde locatie van het bestandssysteem. De volgende strategieën helpen verbindingsfouten te voorkomen:

Gebruik gebruikersspecifieke databasebestanden Geef iedere gebruiker zijn of haar eigen databasebestand, in plaats van één databasebestand te gebruiken voor alle gebruikers die de toepassing op één computer gebruiken. Plaats het bestand in een directory die aan de account van de gebruiker is gekoppeld. Dit kan bijvoorbeeld de opslagmap van de toepassing, de documentenmap van de gebruiker of het bureaublad van de gebruiker zijn.

Houd rekening met verschillende soorten gebruikers Test uw toepassing met verschillende soorten gebruikersaccounts in verschillende besturingssystemen. Ga er niet van uit dat de gebruiker beheerdersrechten heeft voor de computer. Ga er ook niet van uit dat de persoon die de toepassing heeft geïnstalleerd, de gebruiker is die de toepassing uitvoert.

Houd rekening met verschillende bestandslocaties Als u een gebruiker toestaat de opslaglocatie van een databasebestand te bepalen of het te openen bestand te selecteren, moet u rekening houden met de mogelijke bestandslocaties die de gebruikers kunnen selecteren. Bovendien kunt u het best de lijst van mogelijke opslaglocaties (of bronlocaties voor het openen) voor databasebestanden beperken. U kunt bijvoorbeeld gebruikers alleen toestaan bestanden te openen die zich binnen de opslaglocatie van hun gebruikersaccount bevinden.

Als een verbindingsfout optreedt, is dit doorgaans bij de eerste poging om de database te creëren of te openen. Dit betekent dat de gebruiker geen enkele databasegerelateerde bewerking in de toepassing kan uitvoeren. Voor bepaalde soorten fouten, zoals alleen-lezen of toegangsrechtfouten, is een van de mogelijke hersteltechnieken het kopiëren van het databasebestand naar een andere locatie. De toepassing kan het databasebestand kopiëren naar een locatie

waarvoor de gebruiker wel toegangsrechten heeft om bestanden te creëren en te wijzigen, en vervolgens die nieuwe locatie gebruiken.

Syntaxisfouten

Adobe AIR 1.0 of hoger

Een syntaxisfout treedt op als de toepassing een SQL-instructie probeert uit te voeren die niet goed is geformuleerd. Aangezien SQL-instructies voor lokale databases als tekenreeksen zijn gecreëerd, is het niet mogelijk om de SQL-syntaxis tijdens het compileren te controleren. Alle SQL-instructies moeten worden uitgevoerd om hun syntaxis te controleren. Pas de volgende strategieën toe om SQL-syntaxisfouten te voorkomen:

Voer een grondige test uit van alle SQL-instructies Indien mogelijk moet u uw SQL-instructies tijdens de ontwikkeling van uw toepassing apart testen voordat u ze als instructietekst in de toepassingscode codeert. Bovendien wordt u aangeraden een codetestfase in te plannen (zoals het testen per eenheid) om een set van tests te creëren die elke mogelijke optie en codevariatie testen.

Gebruik instructieparameters en vermijd het samenvoegen (dynamisch genereren) van SQL Parameters gebruiken, en dynamisch gegenereerde SQL-instructies vermijden, betekent dat dezelfde SQL-instructietekst wordt gebruikt elke keer dat een instructie wordt uitgevoerd. Hierdoor is het veel eenvoudiger om uw instructies te testen en de mogelijke variaties te beperken. Als u een SQL-instructie dynamisch moet genereren, moet u de dynamische delen van de instructie tot het minimum beperken. Ga ook zorgvuldig te werk bij het valideren van gegevensinvoer om te zorgen dat deze geen syntaxisfouten veroorzaakt.

Een toepassing die van een syntaxisfout moet herstellen, heeft complexe logica nodig om een SQL-instructie te controleren en de syntaxis te corrigeren. Door de volgende richtlijnen voor het voorkomen van syntaxisfouten in acht te nemen, kan uw code mogelijke runtimebronnen van SQL-syntaxisfouten (zoals gebruikersinvoer in een instructie) identificeren. Zorg dat de gebruiker over richtlijnen beschikt als een syntaxisfout optreedt. Geef aan wat de gebruiker moet corrigeren om te zorgen dat de instructie correct wordt uitgevoerd.

Beperkingsfouten

Adobe AIR 1.0 of hoger

Beperkingsfouten treden op wanneer de instructie `INSERT` of `UPDATE` gegevens aan een kolom probeert toe te voegen. De fout treedt op als de nieuwe gegevens een van de vooraf gedefinieerde beperkingen voor de tabel of kolom schenden. U kunt onder andere de volgende beperkingen instellen:

Beperking 'uniek' Geeft aan dat geen enkele rij van de tabel een al bestaande waarde mag bevatten binnen een bepaalde kolom. Of u kunt aangeven dat wanneer meerdere kolommen worden gecombineerd in één beperking, de combinatie van waarden in de desbetreffende kolommen niet mag worden gedupliceerd. Met andere woorden, elke rij moet verschillend zijn op het gebied van de opgegeven unieke kolom(men).

Beperking 'primaire sleutel' Wat de gegevens betreft die een beperking (niet) toestaat, is een beperking 'primaire sleutel' hetzelfde als een beperking 'uniek'.

Beperking 'niet leeg' Geeft aan dat een bepaalde kolom niet de waarde `NULL` mag bevatten, met andere woorden, dat alle rijen in de desbetreffende kolom een waarde moeten hebben.

Beperking 'controle' Biedt u de mogelijkheid om een willekeurige beperking in te stellen voor een of meer tabellen. Een veel voorkomende beperking van het 'controle' is een regel die bepaalt dat de waarde in een kolom zich binnen een bepaald bereik moet bevinden (bijvoorbeeld dat de waarde in een numerieke kolom groter moet zijn dan 0). Een andere veel voorkomende beperking van het type 'controle' definieert relaties tussen kolomwaarden (bijvoorbeeld dat de waarde in een kolom moet verschillen van de waarde in een andere kolom op dezelfde rij).

Beperking 'gegevenstype' De runtime legt het gegevenstype voor de kolomwaarden op en er treedt een fout op als wordt geprobeerd om een waarde op te slaan die niet het juiste type heeft voor de kolom. In vele situaties worden waarden echter omgezet zodat ze het gegevenstype hebben dat is ingesteld voor de kolom. Zie "Werken met gegevenstypen" op pagina 794 voor meer informatie.

De runtime legt geen beperkingen op voor vreemde-sleutelwaarden. Met andere woorden, vreemde-sleutelwaarden hoeven niet overeen te komen met een bestaande primaire-sleutelwaarde.

Naast de vooraf gedefinieerde typen beperkingen ondersteunt de SQL-runtime-engine het gebruik van triggers. Een trigger is te vergelijken met een gebeurtenishandler. Het is een vooraf gedefinieerde instructieset die worden uitgevoerd wanneer zich een bepaalde actie voordoet. U kunt bijvoorbeeld een trigger definiëren die worden geactiveerd wanneer gegevens worden ingevoegd in of verwijderd uit een bepaalde tabel. Een voorbeeld van het gebruik van een trigger is het controleren van wijzigingen in gegevens en het weergeven van een foutmelding als niet wordt voldaan aan de opgegeven voorwaarden. Dit betekent dat een trigger hetzelfde doel kan hebben als een beperking, en dat de strategieën voor het voorkomen en herstellen van beperkingsfouten ook gelden voor fouten die door een trigger worden gegenereerd. De fout-id voor fouten die door een trigger worden gegenereerd, verschilt echter van de fout-id voor beperkingsfouten.

De set van beperkingen die gelden voor een bepaalde tabel wordt gedefinieerd terwijl u een toepassing ontwikkelt. Door goed na te denken bij het definiëren van beperkingen vergemakkelijkt u het voorkomen en herstellen van beperkingsfouten in uw toepassingen. Beperkingsfouten zijn echter moeilijk systematisch te voorspellen en te voorkomen. Het voorspellen is moeilijk omdat beperkingsfouten pas optreden wanneer toepassingsgegevens worden toegevoegd. Beperkingsfouten treden op wanneer gegevens aan een database worden toegevoegd nadat deze is gecreëerd. Deze fouten zijn vaak het resultaat van de relatie tussen nieuwe gegevens en gegevens die al in de database aanwezig zijn. De volgende strategieën kunnen u helpen veel voorkomende beperkingsfouten te voorkomen:

Plan de databasestructuur en de beperkingen zorgvuldig Het doel van beperkingen is toepassingsregels op te leggen en te helpen de integriteit van de databasegegevens te beveiligen. Denk tijdens het plannen van uw toepassing na over de structuur van uw database zodat deze uw toepassing kan ondersteunen. Hierbij identificeert u regels voor de gegevens, zoals of bepaalde waarden vereist zijn, een waarde een standaardinstelling heeft, dubbele waarden zijn toegestaan enzovoort. Deze regels helpen u bij het definiëren van databasebeperkingen.

Geef expliciet kolomnamen op U kunt de instructie `INSERT` gebruiken zonder expliciet de kolommen op te geven waarin waarden zullen worden ingevoegd. Dit is echter een onnodig risico. Door expliciet de kolommen te benoemen waarin waarden zullen worden ingevoegd, kunt u automatisch gegenereerde waarden, kolommen met standaardwaarden en kolommen met de waarde `NULL` gebruiken. Bovendien zorgt u hierdoor dat in alle kolommen van het type `NOT NULL` een expliciete waarde wordt ingevoegd.

Gebruik standaardwaarden Wanneer u de beperking `NOT NULL` voor een kolom opgeeft, geeft u wanneer mogelijk een standaardwaarde in de kolomdefinitie op. Ook toepassingscode kan in standaardwaarden voorzien. Uw code kan bijvoorbeeld controleren of een tekenreeksvariabele de waarde `null` heeft en er een waarde aan toewijzen voordat de variabele wordt gebruikt om de waarde van een instructieparameter in te stellen.

Valideer gebruikersinvoer Controleer gebruikersinvoer voordat de gegevens daadwerkelijk in de database worden ingevoegd om te zorgen dat de invoer aan de beperkingen voldoet, met name in het geval van de beperkingen `NOT NULL` en `CHECK`. De beperking `UNIQUE` is uiteraard moeilijker te controleren omdat een `SELECT`-query moet worden uitgevoerd om te bepalen of de gegevens uniek zijn.

Gebruik triggers U kunt een trigger schrijven die ingevoegde gegevens verifieert (en mogelijk vervangt) of andere acties uitvoert om ongeldige gegevens te corrigeren. Deze verificatie- en correctieacties kunnen beperkingsfouten voorkomen.

In vele opzichten zijn beperkingsfouten moeilijker te voorkomen dan andere fouttypen. Gelukkig zijn er verschillende strategieën om van beperkingsfouten te herstellen zonder dat de toepassing instabiel of onbruikbaar wordt:

Gebruik conflictalgoritmen Als u een beperking voor een kolom definieert en als u de instructie `INSERT` of `UPDATE` creëert, kunt u desgewenst een conflictalgoritme opgeven. Een conflictalgoritme definieert de actie die de database uitvoert als een beperkingsfout optreedt. De database-engine kan verschillende mogelijke acties uitvoeren. De database-engine kan één instructie of een complete transactie beëindigen. De engine kan de fout negeren. Of zelfs oude gegevens verwijderen en deze vervangen door de gegevens die de code probeert op te slaan.

Zie de sectie "ON CONFLICT (conflictalgoritmen)" in "SQL-ondersteuning in lokale databases" op pagina 1176 voor meer informatie.

Geef feedback met een oplossing De set van beperkingen die een bepaalde SQL-opdracht kunnen beïnvloeden, kan van tevoren worden geïdentificeerd. Dit betekent dat u beperkingsfouten die door een instructie kunnen worden veroorzaakt, kunt voorspellen. Op die kennis kunt u toepassingslogica baseren die op een beperkingsfout reageert. Een toepassing bevat bijvoorbeeld een formulier voor het invoeren van nieuwe producten. Als de kolom met de productnaam in de database is gedefinieerd met een beperking van het type `UNIQUE`, kan het invoegen van een nieuwe productrij in de database een beperkingsfout veroorzaken. Daarom wordt de toepassing zo ontwikkeld dat deze van tevoren rekening houdt met een beperkingsfout. Als de fout optreedt, waarschuwt de toepassing de gebruiker dat de opgegeven productnaam al bestaat en dat een andere naam moet worden gekozen. Een andere mogelijke reactie is dat informatie over het andere product met dezelfde naam wordt weergegeven.

Werken met gegevenstypen

Adobe AIR 1.0 of hoger

Bij het creëren van een tabel in een database definieert de SQL-instructie voor het creëren van de tabel het gegevenstype voor elke kolom van de tabel. Het toewijzen van typen is weliswaar niet vereist maar u wordt toch aangeraden het kolomtype expliciet op te geven in uw SQL-instructies `CREATE TABLE`.

Normaliter wordt elk object dat u in een database opslaat met de instructie `INSERT`, weergegeven als een instantie van hetzelfde gegevenstype wanneer u de instructie `SELECT` uitvoert. De opgehaalde waarde kan echter een ander gegevenstype hebben, afhankelijk van de toewijzing van het type aan de databasekolom waarin de waarde is opgeslagen. Als het gegevenstype van een waarde die in een kolom wordt opgeslagen, niet overeenkomt met het type dat aan de kolom is toegewezen, probeert de database de waarde om te zetten in het type dat aan de kolom is toegewezen. Als bijvoorbeeld het gegevenstype `NUMERIC` aan een databasekolom is toegewezen, probeert de database ingevoegde gegevens om te zetten in een numerieke opslagklasse (`INTEGER` of `REAL`) voordat de gegevens worden opgeslagen. De database genereert een foutmelding als de gegevens niet kunnen worden omgezet. Als de tekenreeks "12345" wordt ingevoegd in een kolom van het type `NUMERIC`, zet de database op basis van deze regel de tekenreeks automatisch om in het geheel getal 12345 voordat de waarde in de database wordt opgeslagen. Wanneer de waarde wordt opgehaald met de instructie `SELECT`, wordt deze weergegeven als een instantie van een numeriek gegevenstype (bijvoorbeeld Number) in plaats van een tekenreeksinstantie.

De beste manier om ongewenste omzetting van het gegevenstype te voorkomen, is het in acht nemen van twee regels. Ten eerste: definieer elke kolom met het type dat overeenkomt met het type gegevens dat u wilt opslaan. Ten tweede: voeg alleen waarden in waarvan het gegevenstype overeenkomt met het toegewezen type. Het in acht nemen van deze regels biedt twee voordelen. Wanneer u de gegevens invoegt, worden deze niet onbedoeld omgezet (waardoor mogelijk de bedoelde betekenis verloren zou gaan). Bovendien worden de gegevens bij het ophalen weergegeven met het oorspronkelijke gegevenstype.

Zie "Ondersteuning van gegevenstypen" op pagina 1199 voor meer informatie over de beschikbare typen voor kolomaffiniteit en over het gebruik van gegevenstypen in SQL-instructies.

Synchrone en asynchrone databasebewerkingen gebruiken

Adobe AIR 1.0 of hoger

In de vorige secties zijn veel voorkomende databasebewerkingen besproken, zoals het ophalen, invoegen, updaten en verwijderen van gegevens, evenals het creëren van een databasebestand en tabellen en andere objecten binnen een database. In de voorbeelden hebt u gezien hoe u deze bewerkingen synchroon en asynchroon kunt laten uitvoeren.

Ter herinnering: in asynchrone uitvoeringsmodus geeft u de database-engine de opdracht om een bewerking uit te voeren. De database-engine werkt vervolgens op de achtergrond terwijl de toepassing verderwerkt. Nadat de bewerking is voltooid, verzendt de database-engine een gebeurtenis om u dit te melden. Het belangrijkste voordeel van asynchrone uitvoering is dat de runtime de databasebewerkingen op de achtergrond uitvoert terwijl de hoofdtoepassingscode verder wordt uitgevoerd. Dit is vooral belangrijk wanneer het uitvoeren van de bewerking lang duurt.

In de synchrone uitvoeringsmodus echter worden de bewerkingen niet op de achtergrond uitgevoerd. U geeft de database-engine de opdracht een bewerking uit te voeren. De code wordt op dat punt onderbroken terwijl de database-engine zijn werk doet. Nadat de bewerking is voltooid, gaat de uitvoering verder vanaf de volgende regel van uw code.

Het is niet mogelijk om via één databaseverbinding bepaalde bewerkingen of instructies synchroon en andere asynchroon uit te voeren. U bepaalt welke uitvoeringsmodus (synchroon of asynchroon) voor een SQLConnection wordt gebruikt wanneer u verbinding maakt met de database. Als u `SQLConnection.open()` aanroept, werkt de verbinding in de synchrone uitvoeringsmodus. Roept u `SQLConnection.openAsync()` aan, dan werkt de verbinding in de asynchrone uitvoeringsmodus. Nadat een SQLConnection-instantie met een database is verbonden met behulp van `open()` of `openAsync()`, wordt deze in synchrone of asynchrone uitvoeringsmodus vergrendeld.

Synchrone databasebewerkingen gebruiken

Adobe AIR 1.0 of hoger

Op het gebied van de daadwerkelijke code die u gebruikt om bewerkingen uit te voeren en erop te reageren, is er niet veel verschil tussen de synchrone en asynchrone uitvoeringsmodus. De belangrijkste verschillen tussen beide modi liggen in twee gebieden. Het eerste verschil is de uitvoering van een bewerking die afhankelijk is van een andere bewerking (bijvoorbeeld rijen die het resultaat zijn van `SELECT` of de primaire sleutel van de rij die is toegevoegd door de instructie `INSERT`). Het tweede verschil is de verwerking van fouten.

Code schrijven voor synchrone bewerkingen

Adobe AIR 1.0 of hoger

Het belangrijkste verschil tussen synchrone en asynchrone uitvoering is dat u in de synchrone modus de code als één reeks van stappen schrijft. In de asynchrone modus registreert u gebeurtenislisteners en verdeelt u vaak bewerkingen over verschillende listenermethoden. Als een database in de synchrone uitvoeringsmodus is verbonden, kunt u een reeks van opeenvolgende databasebewerkingen binnen één codeblok laten uitvoeren. Deze techniek wordt in het volgende voorbeeld getoond:

```
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();
```

Zoals u ziet, roept u dezelfde methoden op om databasebewerkingen uit te voeren, ongeacht de modus die u gebruikt (synchroon of asynchroon). De belangrijkste verschillen tussen beide modi zijn de uitvoering van een bewerking die afhankelijk is van een andere bewerking, en de verwerking van fouten.

Bewerkingen uitvoeren die afhankelijk zijn van een andere bewerking

Adobe AIR 1.0 of hoger

In de synchrone uitvoeringsmodus hoeft u geen code te schrijven die op een gebeurtenis wacht om te bepalen wanneer een bewerking is voltooid. U mag ervan uitgaan dat als een bewerking op een bepaalde regel van de code is voltooid, de uitvoering verdergaat met de volgende regel. Dit betekent dat als u een bewerking wilt uitvoeren die afhankelijk is van de voltooiing van een andere bewerking, u gewoon de afhankelijke code onmiddellijk na de bewerking schrijft waarvan deze afhankelijk is. Als u bijvoorbeeld wilt dat een toepassing een transactie start, de instructie `INSERT` uitvoert, de primaire sleutel van de ingevoegde rij ophaalt, die primaire sleutel op een andere rij van een andere tabel invoegt en ten slotte de transactie bevestigt, kunt u de complete code als een reeks van instructies schrijven. In het volgende voorbeeld ziet u het gebruik van deze bewerkingen:

```
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();
```

Fouten verwerken in synchrone uitvoeringsmodus

Adobe AIR 1.0 of hoger

In de synchrone uitvoeringsmodus wacht u niet op een foutgebeurtenis om te bepalen of een bewerking is mislukt. In plaats daarvan plaatst u code die fouten kan veroorzaken in een set codeblokken van het type `try..catch..finally`. Plaats de code die de fout kan genereren in het blok `try`. Schrijf de acties die als reactie op de verschillende fouttypen moeten worden uitgevoerd in aparte blokken van het type `catch`. Plaats eventuele code die u altijd wilt uitvoeren ongeacht het resultaat (bijvoorbeeld het sluiten van een databaseverbinding die u niet meer nodig hebt) in een blok van het type `finally`. In het volgende voorbeeld ziet u hoe u blokken van het type `try..catch..finally` voor foutverwerking gebruikt. Dit voorbeeld is gebaseerd op het vorige voorbeeld maar voegt code voor foutverwerking toe:

```
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();
}
```

De asynchrone uitvoeringsmodus

Adobe AIR 1.0 of hoger

Veel ontwikkelaars gebruiken de asynchrone uitvoeringsmodus liever niet omdat ze denken dat de uitvoering van een SQLStatement-instantie niet kan worden gestart als al een andere SQLStatement wordt uitgevoerd via dezelfde databaseverbinding. Dit is niet juist. U kunt de eigenschap `text` van een SQLStatement-instantie niet wijzigen terwijl de instructie wordt uitgevoerd. Als u echter een aparte SQLStatement-instantie gebruikt voor elke verschillende SQL-instructie die u wilt uitvoeren, kunt u de methode `execute()` van een SQLStatement oproepen terwijl een andere SQLStatement-instantie wordt uitgevoerd, zonder dat een fout optreedt.

Wanneer u databasebewerkingen uitvoert in de asynchrone uitvoeringsmodus, heeft elke databaseverbinding (elke SQLConnection-instantie) een eigen interne wachtrij of lijst van bewerkingen die de verbinding moet uitvoeren. De runtime voert de bewerkingen één voor één uit in de volgorde waarin ze aan de wachtrij zijn toegevoegd. Wanneer u een SQLStatement-instantie creëert en de overeenkomstige methode `execute()` oproept, wordt de desbetreffende bewerking voor het uitvoeren van de instructie toegevoegd aan de wachtrij van de verbinding. Als er momenteel geen bewerking wordt uitgevoerd voor de desbetreffende SQLConnection-instantie, wordt de uitvoering van de instructie op de achtergrond gestart. U kunt bijvoorbeeld binnen hetzelfde codeblok een andere SQLStatement-instantie creëren en ook de methode `execute()` van de desbetreffende methode oproepen. Die tweede bewerking voor het uitvoeren van de instructie wordt na de eerste instructie in de wachtrij geplaatst. Zodra de eerste instructie is voltooid, voert de runtime de volgende bewerking uit de wachtrij uit. De verwerking van daarop volgende bewerkingen uit de wachtrij vindt op de achtergrond plaats, zelfs wanneer de gebeurtenis `result` voor de eerste bewerking wordt verzonden in de hoofdtoepassingscode. In de volgende code wordt deze techniek geïllustreerd:

```
// 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.
```

Er is een belangrijk neveneffect: de database voert automatisch daarop volgende instructies uit de wachtrij uit. Als een instructie afhankelijk is van het resultaat van een andere bewerking, kunt u de instructie niet toevoegen aan de wachtrij (met andere woorden, u kunt de methode `execute()` van de instructie niet oproepen) voordat de eerste bewerking is voltooid. De reden hiervoor is dat nadat u de methode `execute()` van de tweede instructie hebt opgeroepen, u de eigenschap `text` of `parameters` van de instructie niet meer kunt wijzigen. In dat geval kunt u de volgende bewerking pas starten nadat de gebeurtenis is verzonden die aangeeft dat de eerste bewerking is voltooid. Als u bijvoorbeeld een instructie wilt uitvoeren in de context van een transactie, hangt de uitvoering van de instructie af van het openen van de transactie. Nadat u de methode `SQLConnection.begin()` hebt opgeroepen om de transactie te openen, moet u wachten tot de SQLConnection-instantie de gebeurtenis `begin` verzendt. Pas daarna kunt u de methode `execute()` van de SQLStatement-instantie oproepen. In dit voorbeeld is de eenvoudigste manier om de toepassing zo te structureren dat de bewerkingen correct worden uitgevoerd, het creëren van een methode die als listener voor de gebeurtenis `begin` is geregistreerd. De code die de methode `SQLStatement.execute()` oproept, wordt binnen die listenermethode geplaatst.

Codering gebruiken met SQL-databases

Adobe AIR 1.5 of hoger

Alle Adobe AIR-toepassingen delen dezelfde lokale database-engine. Elke AIR-toepassing kan daarom verbinding maken met en gegevens lezen van en schrijven naar een niet-gecodeerd databasebestand. Vanaf Adobe AIR 1.5 is AIR ook voorzien van de mogelijkheid om gecodeerde databasebestanden te maken en hiermee verbinding te maken. Wanneer u een gecodeerde database gebruikt, moet een toepassing de correcte coderingssleutel opgeven om verbinding te kunnen maken met deze database. Als een onjuiste of geen sleutel wordt opgegeven, kan de toepassing geen verbinding maken met de database. De toepassing kan dan ook geen gegevens van de database lezen of gegevens naar de database schrijven of wijzigen.

Als u een gecodeerde database wilt gebruiken, moet u de desbetreffende gecodeerde database maken. Bij een bestaande gecodeerde database kunt u een verbinding tot stand brengen. U kunt ook de coderingssleutel van een gecodeerde database wijzigen. De methoden voor het werken met een gecodeerde database zijn bijna hetzelfde als die voor het werken met een niet-gecodeerde. Het enige verschil is dat u de gecodeerde database moet maken en er verbinding mee tot stand moet brengen. Vooral de uitvoering van SQL-instructies is hetzelfde, of de database nu wel of niet gecodeerd is.

Gebruik van gecodeerde databases

Adobe AIR 1.5 of hoger

Codering is nuttig als u de toegang tot de informatie in een database wilt beperken. U kunt de databasecoderingsfunctie van Adobe AIR gebruiken voor verschillende doeleinden. Hieronder volgen enige voorbeelden van situaties waarbij u een gecodeerde database kunt gebruiken:

- Een alleen-lezen cache van privétoepassingsgegevens die u hebt gedownload van een server
- Een lokale opslag van privétoepassingsgegevens die zijn gesynchroniseerd met een server (gegevens worden verzonden naar en geladen van de server)
- Gecodeerde bestanden gebruikt als bestandsindeling voor documenten die zijn gemaakt door en bewerkt door de toepassing. Dit kunnen privébestanden van een bepaalde gebruiker zijn, of ze kunnen worden gedeeld door alle gebruikers van de toepassing.
- Elk ander gebruik van een lokale gegevensopslag, zoals beschreven in "Toepassingen voor lokale SQL-databases" op pagina 754, waarbij de gegevens niet toegankelijk mogen zijn voor andere gebruikers die toegang hebben tot de computer of de databasebestanden.

Inzicht in de reden waarom u een gecodeerde database wilt gebruiken kan u helpen met de besluitvorming betreffende de architectuur van de toepassing. Dit is vooral van belang voor de manier waarop uw toepassing de coderingssleutel voor de database maakt, verkrijgt en opslaat. Raadpleeg "Overwegingen bij het gebruik van codering voor een database" op pagina 804 voor meer informatie.

Behalve een gecodeerde database kunt u ook een gecodeerde lokale opslag gebruiken om gevoelige gegevens privé te houden. Bij de gecodeerde lokale opslag slaat u één enkele ByteArray-waarde op met behulp van een tekenreekssleutel. Alleen de AIR-toepassing die de waarde heeft opgeslagen, kan deze benaderen. Dit kan ook alleen op de computer waarop deze is opgeslagen. Met de gecodeerde lokale opslag is het niet nodig om uw eigen coderingssleutel te maken. Vanwege deze redenen is de gecodeerde lokale opslag het meest geschikt voor het opslaan van één waarde of een set waarden die gemakkelijk kunnen worden gecodeerd in een ByteArray. Een gecodeerde database is het meest geschikt voor grotere gegevenssets waarbij gestructureerde gegevensopslag en het uitvoeren van query's gewenst zijn. Zie "Lokale gegevens gecodeerd opslaan" op pagina 750 voor meer informatie over het beveiligd opslaan van lokale gegevens.

Een gecodeerde database maken

Adobe AIR 1.5 of hoger

Om een gecodeerde database te gebruiken, moet het databasebestand zijn gecodeerd wanneer het wordt gemaakt. Een niet-gecodeerde database kan later niet alsnog worden gecodeerd. Ook kan de codering van een gecodeerde database later niet worden opgeheven. U kunt desgewenst de coderingssleutel van een gecodeerde database wijzigen. Zie "De coderingssleutel van een database wijzigen" op pagina 803 voor meer informatie. Als u een bestaande database hebt die niet gecodeerd is en u gebruik wilt maken van databasecodering, kunt u een nieuwe gecodeerde database maken en de bestaande tabelstructuur en gegevens kopiëren naar die nieuwe database.

Het maken van een gecodeerde database gaat op bijna dezelfde manier als het maken van een niet-gecodeerde database; dit wordt beschreven in "Database creëren" op pagina 759. Eerst maakt u een SQLConnection-instantie die de verbinding met de database vertegenwoordigt. U maakt de database door de methode `open()` of de methode `openAsync()` van het object SQLConnection aan te roepen. Geef hierbij voor de databaselocatie een bestand op dat nog niet bestaat. Wanneer u een gecodeerde database maakt, is het enige verschil dat u een waarde opgeeft voor de parameter `encryptionKey` (de vijfde parameter van de methode `open()` en de zesde parameter van de methode `openAsync()`).

Een geldige waarde voor de parameter `encryptionKey` is een ByteArray-object dat precies 16 bytes bevat.

In de volgende voorbeelden wordt getoond hoe u een gecodeerde database kunt maken. Voor het gemak is in deze voorbeeld de coderingssleutel vastgelegd in de toepassingscode. Dit is echter sterk af te raden omdat het niet veilig is.

```
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);
```

Een voorbeeld van een aanbevolen manier om een coderingssleutel te genereren vindt u in "Voorbeeld: Een coderingssleutel genereren en gebruiken" op pagina 805.

Verbinding maken met een gecodeerde database

Adobe AIR 1.5 of hoger

De procedure voor het tot stand brengen van een verbinding met een gecodeerde database is vergelijkbaar met die voor het tot stand brengen van een verbinding met een niet-gecodeerde database. Die procedure wordt uitgebreid beschreven in "Verbinden met een database" op pagina 767. U gebruikt de methode `open()` voor het openen van een verbinding in de synchrone uitvoeringsmodus en de methode `openAsync()` voor het openen van een verbinding in de asynchrone uitvoeringsmodus. Het enige verschil is dat u voor het openen van een gecodeerde database de correcte waarde opgeeft voor de parameter `encryptionKey` (de vijfde parameter van de methode `open()` en de zesde parameter van de methode `openAsync()`).

Als de opgegeven coderingssleutel niet correct is, treedt er een fout op. Bij de methode `open()` wordt een SQLError-uitzondering gegenereerd. Bij de methode `openAsync()` verzendt het object SQLConnection een SQLErrorEvent-gebeurtenis, waarvan de eigenschap `error` een SQLError-object bevat. In beide gevallen heeft van het object SQLError dat door de uitzondering wordt gegenereerd, de eigenschap `errorID` de waarde 3138. Die fout-ID correspondeert met het foutbericht dat het geopende bestand geen databasebestand is.

In het volgende voorbeeld wordt gedemonstreerd hoe u een gecodeerde database opent in de asynchrone uitvoeringsmodus. Voor het gemak is in dit voorbeeld de coderingssleutel hard-gecodeerd in de toepassingscode. Dit is echter sterk af te raden omdat het niet veilig is.

```
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);
    }
}
```

In het volgende voorbeeld wordt gedemonstreerd hoe u een gecodeerde database opent in de synchrone uitvoeringsmodus. Voor het gemak is in dit voorbeeld de coderingssleutel hard-gecodeerd in de toepassingscode. Dit is echter sterk af te raden omdat het niet veilig is.

```
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);
    }
}
```

Een voorbeeld van een aanbevolen manier om een coderingssleutel te genereren vindt u in "Voorbeeld: Een coderingssleutel genereren en gebruiken" op pagina 805.

De coderingssleutel van een database wijzigen

Adobe AIR 1.5 of hoger

Wanneer een database gecodeerd is, kunt u de coderingssleutel ervan later wijzigen. Als u de coderingssleutel van een database wilt wijzigen, opent u eerst een verbinding met de database door een SQLConnection-instantie te maken en de methode `open()` of `openAsync()` ervan aan te roepen. Als de database verbonden is, roept u de methode `reencrypt()` aan en geeft u de nieuwe coderingssleutel daarbij door als argument.

Zoals bij de meeste bewerkingen op databases, varieert het gedrag van de methode `reencrypt()`; dit is ervan afhankelijk of de databaseverbinding gebruikmaakt van een synchrone dan wel een asynchrone uitvoeringsmodus. Als u de methode `open()` gebruikt om verbinding te maken met de database, worden de `reencrypt()`-bewerkingen synchroon uitgevoerd. Nadat de bewerking is voltooid, gaat de uitvoering verder vanaf de volgende regel code:

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

Als de databaseverbinding echter wordt geopend met de methode `openAsync()`, is de werking van `reencrypt()` asynchroon. De verandering van de coderingssleutel begint met het aanroepen van `reencrypt()`. Wanneer deze bewerking is voltooid, verzendt het object SQLConnection een gebeurtenis `reencrypt`. U gebruikt een gebeurtenislistener om te bepalen wanneer het proces reencrypt voltooid is:

```
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
}
```

De bewerking `reencrypt()` wordt uitgevoerd in zijn eigen transactie. Als de bewerking wordt onderbroken of mislukt (bijvoorbeeld wanneer de toepassing wordt gesloten voordat de bewerking is voltooid), wordt de transactie teruggedraaid. In dat geval is de oorspronkelijke coderingssleutel nog steeds de sleutel voor de database.

De methode `reencrypt()` kan niet worden gebruikt om de codering van een database te verwijderen. Als de waarde `null` of een coderingssleutel die niet bestaat uit een 16-byte ByteArray, wordt doorgegeven aan de methode `reencrypt()`, treedt er een fout op.

Overwegingen bij het gebruik van codering voor een database

Adobe AIR 1.5 of hoger

In de sectie "Gebruik van gecodeerde databases" op pagina 800 wordt een aantal omstandigheden beschreven waaronder u gebruik kunt maken van gecodeerde databases. De gebruikscenario's van verschillende toepassingen (inclusief deze en andere scenario's) hebben uiteraard verschillende privacyvereisten. De mate waarin de gegevens in een database privé blijven, wordt voor een groot deel bepaald door de manier waarop u gebruikmaakt van codering in uw toepassing. Als u bijvoorbeeld een gecodeerde database gebruikt om persoonlijke gegevens privé te houden, zodat zelfs andere gebruikers van dezelfde computer er niet bij kunnen, heeft de database van iedere gebruiker een eigen coderingssleutel nodig. Voor optimale beveiliging kan uw toepassing de sleutel genereren aan de hand van een wachtwoord dat door de gebruiker wordt ingevoerd. Als de coderingssleutel wordt gebaseerd op een wachtwoord, kan zelfs een andere gebruiker die in staat is zich voor te doen als gebruiker van deze account op de computer, geen toegang krijgen tot deze gegevens. Aan de andere kant van het privacyspectrum vindt u databasebestanden die kunnen worden gelezen door iedere gebruiker van uw toepassing, maar niet door andere toepassingen. In dat geval moet ieder geïnstalleerd exemplaar van de toepassing toegang hebben tot een gedeelde coderingssleutel.

U kunt het ontwerp van uw toepassing en vooral de methode die wordt gebruikt om de coderingssleutel te genereren, aanpassen aan het privacyniveau dat u wilt gebruiken voor uw toepassingsgegevens. Hieronder volgt een lijst met suggesties voor verschillende privacyniveaus voor gegevens:

- Als u een database toegankelijk wil maken voor iedere gebruiker die toegang heeft tot de toepassing op iedere computer, gebruikt u één sleutel die beschikbaar is voor alle exemplaren van de toepassing. De eerste keer dat een toepassing wordt uitgevoerd, kan deze de gedeelde coderingssleutel bijvoorbeeld downloaden vanaf een server met gebruikmaking van een veilig protocol zoals SSL. De toepassing kan de sleutel dan opslaan in de gecodeerde lokale opslag voor toekomstig gebruik. Als alternatief kunt u de gegevens per gebruiker coderen op de computer en de gegevens synchroniseren met een externe gegevensopslag zoals een server, zodat de gegevens overdraagbaar zijn.
- Als u een database voor slechts één gebruiker toegankelijk wilt maken op alle computers, genereert u de coderingssleutel aan de hand van een geheim dat alleen de gebruiker kent (bijvoorbeeld een wachtwoord). U moet dan absoluut niet gebruikmaken van een waarde die is gekoppeld aan een bepaalde computer (bijvoorbeeld een waarde die is opgeslagen in de gecodeerde lokale opslag) om de sleutel te genereren. Als alternatief kunt u de gegevens per gebruiker coderen op de computer en de gegevens synchroniseren met een externe gegevensopslag zoals een server, zodat de gegevens overdraagbaar zijn.

- Als een database toegankelijk moet zijn voor één gebruiker op één computer, genereert u de sleutel aan de hand van een wachtwoord en een gegenereerde salt. Een voorbeeld van deze methode vindt u in "Voorbeeld: Een coderingssleutel genereren en gebruiken" op pagina 805.

Hieronder volgt een aantal extra beveiligingsoverwegingen waarmee u rekening moet houden als u een toepassing ontwerpt die gebruikmaakt van een gecodeerde database:

- Een systeem is zo veilig als de zwakste schakel. Als u gebruikmaakt van een door de gebruiker ingevoerd wachtwoord om een coderingssleutel te genereren, is het aan te raden beperkingen met betrekking tot de lengte en de complexiteit van het wachtwoord op te geven. Een kort wachtwoord met eenvoudige tekens is gemakkelijk te raden.
- De broncode van een AIR-toepassing wordt op de computer van een gebruiker opgeslagen als onbewerkte tekst (voor HTML-inhoud) of met een gemakkelijk te decompileren binaire indeling (voor SWF-inhoud). Omdat de broncode toegankelijk is, moet u rekening houden met de volgende twee punten:
  - Maak nooit gebruik van een hard-gecodeerde sleutel in de broncode
  - Ga er altijd van uit dat de methode die wordt gebruikt om een coderingssleutel te genereren (bijvoorbeeld een willekeurige tekengenerator of een bepaald hashalgoritme) gemakkelijk kan worden bepaald door een aanvaller
- AIR-databasecodering maakt gebruik van de Advanced Encryption Standard (AES) met Counter with CBC-MAC (CCM)-modus. Deze codering is alleen veilig als een door de gebruiker ingevoerde sleutel wordt gecombineerd met een salt-waarde. Een voorbeeld van deze methode vindt u in "Voorbeeld: Een coderingssleutel genereren en gebruiken" op pagina 805.
- Wanneer u ervoor kiest om een database te coderen, worden alle schijfbestanden gecodeerd die door de database-engine samen met die database worden gebruikt. De database-engine slaat echter bepaalde gegevens tijdelijk op in een geheugencache om de lees- en schrijfprestaties bij transacties te verbeteren. Gegevens die aanwezig zijn in het geheugen, zijn niet gecodeerd. Als een aanvaller toegang kan krijgen tot het geheugen dat wordt gebruikt door een AIR-toepassing, bijvoorbeeld door middel van een debugger, zijn de gegevens in een database die nu open en niet-gecodeerd is, beschikbaar.

Voorbeeld: Een coderingssleutel genereren en gebruiken

Adobe AIR 1.5 of hoger

In deze voorbeeldtoepassing wordt een methode gedemonstreerd om een coderingssleutel te genereren. Deze toepassing is speciaal ontworpen om het maximale privacy- en beveiligingsniveau te bieden voor de gegevens van de gebruikers. Eén belangrijk aspect van het beveiligen van privégegevens is dat de gebruiker een wachtwoord moet invoeren telkens wanneer de toepassing verbinding met de database maakt. Zoals in dit voorbeeld wordt geïllustreerd, betekent dit dat een toepassing die een dergelijk privacyniveau vereist, de databasecoderingssleutel nooit rechtstreeks mag opslaan.

De toepassing bestaat uit twee delen: een ActionScript-klasse die een coderingssleutel genereert (de klasse EncryptionKeyGenerator) en een basisgebruikersinterface die demonstreert hoe die klasse moet worden gebruikt. De volledige broncode vindt u in "Complete voorbeeldcode voor het genereren en gebruiken van een coderingssleutel" op pagina 808.

De klasse EncryptionKeyGenerator gebruiken om een veilige coderingssleutel te verkrijgen.

Adobe AIR 1.5 of hoger

U kunt de klasse EncryptionKeyGenerator gebruiken in uw toepassing zonder dat u alle details van deze klasse moet kennen. Zie "De klasse EncryptionKeyGenerator" op pagina 812 als u wilt weten hoe de klasse een versleutelingssleutel voor een database genereert.

Ga als volgt te werk om de klasse EncryptionKeyGenerator in uw toepassing te gebruiken:

1. Download de EncryptionKeyGenerator-klasse als broncode of een gecompileerd SWC-bestand. De klasse EncryptionKeyGenerator is opgenomen in het as3corelib-project (open-source ActionScript 3.0 core library). U kunt het as3corelib-pakket, inclusief broncode en documentatie downloaden. U kunt de SWC- of broncodebestanden ook downloaden van de projectpagina.
2. Plaats de broncode voor de klasse EncryptionKeyGenerator (of de as3corelib-SWC) op een locatie waar de broncode van de toepassing deze kan vinden.
3. Voeg aan de broncode van uw toepassing een `import`-instructie voor de klasse EncryptionKeyGenerator toe.

   ```
   import com.adobe.air.crypto.EncryptionKeyGenerator;
   ```

4. Vóór het punt waarop de code de database maakt of een verbinding met de database opent, voegt u code toe om een EncryptionKeyGenerator-instantie te maken. Daartoe roept u de `EncryptionKeyGenerator()`-constructor aan.

   ```
   var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
   ```

5. Vraag een wachtwoord aan de gebruiker:

   ```
   var password:String = passwordInput.text;

   if (!keyGenerator.validateStrongPassword(password))
   {
       // display an error message
       return;
   }
   ```

   De EncryptionKeyGenerator-instantie gebruikt dit wachtwoord als de basis voor de coderingssleutel (weergegeven in de volgende stap). De EncryptionKeyGenerator-instantie test het wachtwoord aan de hand van bepaalde vereisten voor sterke wachtwoorden. Als de validatie mislukt, treedt een fout op. Zoals in de voorbeeldcode wordt geïllustreerd, kunt u het wachtwoord op voorhand controleren door de methode `validateStrongPassword()` van het EncryptionKeyGenerator-object aan te roepen. Op die manier kunt u bepalen of het wachtwoord aan de minimale vereisten voor een sterk wachtwoord voldoet, en kunt u een fout vermijden.

6. Genereer de coderingssleutel vanuit het wachtwoord:

   ```
   var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
   ```

   De methode `getEncryptionKey()` genereert en retourneert de coderingssleutel (een ByteArray van 16 bytes). U kunt de coderingssleutel gebruiken om de nieuwe gecodeerde database te maken of u kunt de bestaande openen.

   De methode `getEncryptionKey()` heeft één verplichte parameter, namelijk het wachtwoord dat in stap 5 is verkregen.

   Opmerking: *Om het hoogste niveau voor beveiliging en privacy van de gegevens te behouden, moet een toepassing de gebruiker om het wachtwoord vragen, telkens wanneer de toepassing verbinding met de database maakt. Sla het wachtwoord of de coderingssleutel van de toepassing niet rechtstreeks op. Dit levert beveiligingsrisico's op. In plaats daarvan moet een toepassing -zoals in dit voorbeeld wordt geïllustreerd - telkens opnieuw de dezelfde techniek gebruiken om de coderingssleutel uit het wachtwoord te halen, dus zowel wanneer de gecodeerde database wordt gemaakt als wanneer later verbinding met de database wordt gemaakt.*

De methode `getEncryptionKey()` accepteert ook een tweede (optionele) parameter, de parameter `overrideSaltELSKey`. De EncryptionKeyGenerator maakt een willekeurige waarde (een zogenaamde *salt*) die als deel van de coderingssleutel wordt gebruikt. Om de coderingssleutel opnieuw te kunnen maken, wordt de salt-waarde opgeslagen in de ELS (Encrypted Local Store) van de AIT-toepassing. Standaard gebruikt de klasse EncryptionKeyGenerator een bepaalde tekenreeks als de ELS-sleutel. Hoewel onwaarschijnlijk, is het mogelijk dat de sleutel een conflict kan opleveren met een andere sleutel die in de toepassing wordt gebruikt. In plaats van de standaardsleutel te gebruiken, kunt u ook uw eigen ELS-sleutel opgeven. In dat geval geeft u een aangepaste sleutel op door deze als tweede `getEncryptionKey()`-parameter door te geven, zoals hier wordt geïllustreerd:

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 De database maken of openen

Met de coderingssleutel die door de methode `getEncryptionKey()` wordt geretourneerd, kan uw code een nieuwe gecodeerde database maken of proberen om verbinding te maken met de bestaande gecodeerde database. In beide gevallen gebruikt u de methode `open()` of `openAsync()` van de klasse SQLConnection zoals wordt beschreven in "Een gecodeerde database maken" op pagina 801 en "Verbinding maken met een gecodeerde database" op pagina 801.

In dit voorbeeld is de toepassing ontworpen om de database te openen in de asynchrone uitvoeringsmodus. De code stelt de juiste gebeurtenislisteners in en roept de methode `openAsync()` aan, waarbij de coderingssleutel wordt doorgegeven als het laatste argument:

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

In de listenermethoden verwijdert de code de gebeurtenislistener-registraties. Vervolgens wordt een statusbericht weergegeven waarin wordt aangegeven of de database is gemaakt, geopend of dat er een fout is opgetreden. Het belangrijkste gedeelte van deze gebeurtenishandlers zit in de methode `openError()`. In die methode controleert een `if`-instructie of de database bestaat (wat betekent dat er wordt geprobeerd verbinding te maken met een bestaande database) en of de fout-id overeenkomt met de constante `EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID`. Als beide voorwaarden waar zijn, betekent dit waarschijnlijk dat de gebruiker een verkeerd wachtwoord heeft ingevoerd. (Het kan ook betekenen dat het opgegeven bestand helemaal geen databasebestand is.) Hier volgt de code die de fout-id controleert:

```
if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

Voor de volledige code van de voorbeeldgebeurtenislisteners raadpleegt u "Complete voorbeeldcode voor het genereren en gebruiken van een coderingssleutel" op pagina 808.

Complete voorbeeldcode voor het genereren en gebruiken van een coderingssleutel

Adobe AIR 1.5 of hoger

Hier volgt de volledige code voor de voorbeeldtoepassing 'Een coderingssleutel genereren en gebruiken'. De code bestaat uit twee delen.

Het voorbeeld gebruikt de klasse EncryptionKeyGenerator om een coderingssleutel te maken op basis van een wachtwoord. De klasse EncryptionKeyGenerator is opgenomen in het as3corelib-project (open-source ActionScript 3.0 core library). U kunt het as3corelib-pakket, inclusief broncode en documentatie downloaden. U kunt de SWC- of broncodebestanden ook downloaden van de projectpagina.

Flex-voorbeeld

Het MXML-bestand van de toepassing bevat de broncode voor een eenvoudige toepassing die een verbinding met een gecodeerde database maakt of opent:

```
<?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>
```

Flash Professional-voorbeeld

Het FLA-bestand van de toepassing bevat de broncode voor een eenvoudige toepassing die een verbinding met een gecodeerde database maakt of opent. Het FLA-bestand heeft vier onderdelen in het werkgebied geplaatst:

Instantienaam	Type onderdeel	Beschrijving
`instructies`	Label	Bevat de instructies die aan de gebruiker worden gegeven
`passwordInput`	TextInput	Invoerveld waarin de gebruiker het wachtwoord typt
`openButton`	Button	Knop waarop de gebruiker klikt nadat het wachtwoord is ingevoerd
`statusMsg`	Label	Geeft statusberichten (geslaagd of niet geslaagd) weer

De code voor de toepassing wordt gedefinieerd in een hoofdframe in frame 1 van de hoofdtijdlijn. Hier volgt de code voor de toepassing:

```
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.";
    }
}
```

De klasse EncryptionKeyGenerator

Adobe AIR 1.5 of hoger

Het is niet nodig dat u precies begrijpt hoe de klasse EncryptionKeyGenerator werkt om deze te gebruiken om een veilige coderingssleutel voor uw toepassingsdatabase te maken. Het gebruik van de klasse wordt uitgelegd in "De klasse EncryptionKeyGenerator gebruiken om een veilige coderingssleutel te verkrijgen." op pagina 806. Het is misschien echter handig om te begrijpen welke technieken de klasse gebruikt. Misschien wilt de klasse bijvoorbeeld aanpassen, of bepaalde technieken van deze klasse gebruiken voor situaties waarin een ander niveau van gegevensprivacy is gewenst.

De klasse EncryptionKeyGenerator is opgenomen in het as3corelib-project (open-source ActionScript 3.0 core library). U kunt het as3corelib-pakket inclusief broncode en documentatie downloaden. U kunt ook de broncode op de projectsite bekijken of deze downloaden om tijdens de uitleg te volgen.

Wanneer code een EncryptionKeyGenerator-exemplaar maakt en de methode `getEncryptionKey()` aanroept, moeten enkele stappen worden uitgevoerd om ervoor te zorgen dat alleen de juiste gebruiker toegang heeft tot de gegevens. Het proces voor het genereren van een coderingssleutel vanuit een door de gebruiker ingevoerd wachtwoord voordat de database wordt gemaakt, is precies hetzelfde als wanneer de coderingssleutel opnieuw moet worden gemaakt om de database te openen.

Een sterk wachtwoord verkrijgen en valideren

Adobe AIR 1.5 of hoger

Wanneer code de methode `getEncryptionKey()` aanroept, wordt een wachtwoord als parameter doorgegeven. Dit wachtwoord wordt gebruikt als de basis voor de coderingssleutel. Door gebruik te maken van informatie die alleen bekend is aan de gebruiker, bent u er met dit ontwerp zeker van dat alleen de gebruiker die het wachtwoord kent, toegang kan krijgen tot de gegevens in de database. Zelfs als een aanvaller toegang kan krijgen tot de account van die gebruiker op de computer, kan de aanvaller de database alleen openen als hij beschikt over het wachtwoord. Voor maximale veiligheid slaat de toepassing het wachtwoord nooit op.

De code van een toepassing maakt een EncryptionKeyGenerator-instantie en roept de methode `getEncryptionKey()` aan, en geeft een door de gebruiker ingevoerd wachtwoord op als argument (het wachtwoord `variable` in dit voorbeeld):

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

Wanneer de methode `getEncryptionKey()` wordt aangeroepen, controleert de klasse EncryptionKeyGenerator eerst het door de gebruiker ingevoerde wachtwoord om na te gaan of het een sterk wachtwoord is. Het wachtwoord voor de EncryptionKeyGenerator-klasse moet 8 tot 32 tekens lang zijn. Het wachtwoord moet zowel hoofdletters als kleine letters bevatten en ten minste één cijfer of symbool.

De reguliere expressie waarmee dit patroon wordt gecontroleerd, is gedefinieerd als een constante met de naam `STRONG_PASSWORD_PATTERN`:

```
private static const STRONG_PASSWORD_PATTERN:RegExp =
/(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

De code waarmee het wachtwoord wordt gecontroleerd, bevindt zich in de methode `validateStrongPassword()` van de klasse EncryptionKeyGenerator. De code is als volgt:

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

De methode `getEncryptionKey()` roept de methode `validateStrongPassword()` van de EncryptionKeyGenerator-klasse intern aan en genereert een uitzondering als het wachtwoord ongeldig is. De methode `validateStrongPassword()` is een publieke methode zodat de toepassingscode een wachtwoord kan controleren zonder de methode `getEncryptionKey()` aan te roepen. Hierdoor wordt vermeden dat een fout optreedt.

Het wachtwoord uitbreiden tot 256 bits

Adobe AIR 1.5 of hoger

Verderop in het proces moet het wachtwoord 256 bits lang zijn. De code vraagt niet iedere gebruiker een wachtwoord in te voeren dat precies 256 bits (32 tekens) lang is, maar creëert een langer wachtwoord door de tekens van het wachtwoord te herhalen.

De methode `getEncryptionKey()` roept de methode `concatenatePassword()` aan om het lange wachtwoord te maken.

```
var concatenatedPassword:String = concatenatePassword(password);
```

Hier volgt de code voor de methode `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;
}
```

Als het wachtwoord korter is dan 256 bits, voegt de code het wachtwoord nogmaals toe om er 256 bits van te maken. Als de lengte niet precies klopt, wordt het laatste stuk afgekapt zodat het wachtwoord precies 256 bits lang is.

Een salt-waarde van 256 bits genereren of ophalen

Adobe AIR 1.5 of hoger

De volgende stap is het verkrijgen van een salt-waarde van 256 bits die in een latere stap wordt gecombineerd met het wachtwoord. Een *salt* is een willekeurige waarde die wordt toegevoegd aan of gecombineerd met een door de gebruiker ingevoerde waarde om een wachtwoord te vormen. Als u een salt gebruikt met een wachtwoord, weet u zeker dat zelfs als de gebruiker een bestaand woord of veelgebruikte term als wachtwoord gebruikt, de combinatie van wachtwoord met salt die het systeem gebruikt, een willekeurige waarde is. Deze willekeurigheid helpt een woordenlijstaanval te voorkomen, waarbij een hacker een lijst met woorden gebruikt om een wachtwoord te raden. Verder wordt door het genereren van de salt-waarde en het opslaan daarvan in de gecodeerde lokale opslag, de salt-waarde gekoppeld aan de account van de gebruiker op de computer waarop de database zich bevindt.

Als de toepassing de methode `getEncryptionKey()` de eerste keer aanroept, maakt de code een willekeurige salt-waarde van 256 bits. In het vervolg laadt de code de waarde uit de gecodeerde lokale opslag.

De salt wordt opgeslagen in een variabele met de naam `salt`. De code bepaalt of er al een salt is gemaakt. Dit gebeurt door de salt te laden vanaf een beveiligde opslagplaats voor lokale gegevens:

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

Als de code een nieuwe salt-waarde maakt, genereert de methode `makeSalt()` een willekeurige waarde van 256 bits. Aangezien deze waarde uiteindelijk wordt opgeslagen in de gecodeerde lokale opslag, wordt de waarde gegenereerd als ByteArray-object. De methode `makeSalt()` gebruikt de methode `Math.random()` om een willekeurige waarde te genereren. De methode `Math.random()` kan niet 256 bits tegelijk genereren. In plaats daarvan gebruikt de code een lus om `Math.random()` acht keer aan te roepen. Iedere keer wordt een willekeurige uint-waarde tussen 0 en 4294967295 (de maximale uint-waarde) gegenereerd. De uint-waarde wordt voor het gemak gebruikt, aangezien een uint precies 32 bits gebruikt. Door acht uint-waarden naar de ByteArray te schrijven wordt een waarde van 256 bits gegenereerd. Hieronder volgt de code voor de methode `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;
}
```

Wanneer de code de salt opslaat in de ELS (Encrypted Local Store) of de salt uit de ELS ophaalt, heeft deze een String-sleutel nodig waarin de salt wordt opgeslagen. De salt-waarde kan niet worden opgehaald zonder de sleutel te kennen. In dat geval kan de coderingssleutel niet opnieuw worden gemaakt wanneer de database opnieuw moet worden geopend. Standaard gebruikt de EncryptionKeyGenerator een op voorhand gedefinieerde ELS-sleutel die is gedefinieerd in de constante `SALT_ELS_KEY`. In plaats van de standaardsleutel te gebruiken, kan de toepassingscode ook een ELS-sleutel opgeven die moet worden gebruikt in de aanroep van de methode `getEncryptionKey()`. De standaardsleutel of de in de toepassing opgegeven ELS-sleutels wordt opgeslagen in een variabele met de naam `saltKey`. Deze variabele wordt gebruikt om `EncryptedLocalStore.setItem()` en `EncryptedLocalStore.getItem()` aan te roepen, zoals eerder werd getoond.

Het 256-bits wachtwoord combineren met de salt met behulp van de XOR-operator

Adobe AIR 1.5 of hoger

De code heeft nu een wachtwoord van 256 bits en een salt-waarde van 256 bits. Vervolgens wordt een bitsgewijze XOR-bewerking gebruikt om de salt en het samengevoegde wachtwoord in één waarde te gebruiken. In feite wordt met deze techniek een wachtwoord van 256 bits gemaakt bestaande uit tekens uit het hele bereik van mogelijke tekens. Dit principe is waar, ook al bestaat het ingevoerde wachtwoord hoogstwaarschijnlijk voornamelijk uit alfanumerieke tekens. Deze extra willekeurigheid biedt het voordeel dat de set mogelijke wachtwoorden groot is zonder dat de gebruiker een lang en complex wachtwoord moet invoeren.

Het resultaat van de XOR-bewerking wordt opgeslagen in de variabele `unhashedKey`. De bitsgewijze XOR voor de twee waarden wordt in werkelijkheid uitgevoerd in de methode `xorBytes()`:

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

De bitsgewijze XOR-operator (^) gebruikt twee uint-waarden en geeft als resultaat een uint-waarde. (Een uint-waarde bevat 32 bits.) De ingevoerde waarden die als argumenten worden doorgegeven aan de methode `xorBytes()`, bestaan uit een tekenreeks (het wachtwoord) en een ByteArray (de salt). De code gebruikt een lus om 32 bits per keer uit ieder invoergegeven te extraheren, die dan worden gecombineerd met gebruikmaking van de XOR-operator.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

Binnen de lus worden de eerste 32 bits (4 bytes) uit de parameter `passwordString` gehaald. Deze bits worden geëxtraheerd en vervolgens geconverteerd naar een (`o1`) met een proces dat uit twee delen bestaat. Eerst haalt de methode `charCodeAt()` de numerieke waarde van ieder teken op. Vervolgens wordt deze waarde naar de juiste positie in de uint verschoven met behulp van de bitsgewijze operator voor verschuiven naar links (`<<`). De verschoven waarde wordt toegevoegd aan `o1`. Het eerste teken (`i`) wordt bijvoorbeeld de eerste 8 bits door de operator voor bitsgewijs naar links verplaatsen (`<<`) te gebruiken om de bits 24 bits naar links te verplaatsen en die waarde toe te wijzen aan `o1`. Het tweede teken (`i + 1`) wordt de tweede 8 bits door de waarde ervan 16 bits naar links te verplaatsen en het resultaat toe te voegen aan `o1`. De waarden voor het derde en het vierde teken worden op dezelfde manier toegevoegd.

```
        // ...

        // 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);

        // ...
```

De variabele `o1` bevat nu 32 bits uit de parameter `passwordString`. Vervolgens worden 32 bits geëxtraheerd uit de parameter `salt` door de methode `readUnsignedInt()` ervan aan te roepen. De 32 bits worden opgeslagen in de uint-variabele `o2`.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

Tot slot worden de twee 32-bits waarden gecombineerd met de operator XOR en wordt het resultaat weggeschreven naar een ByteArray met de naam `result`.

```
            // ...

            var xor:uint = o1 ^ o2;
            result.writeUnsignedInt(xor);
            // ...
```

Als de lus is voltooid, wordt de ByteArray met het XOR-resultaat geretourneerd.

```
            // ...
        }

        return result;
}
```

De sleutel hashen

Adobe AIR 1.5 of hoger

Wanneer het samengevoegde wachtwoord en de salt zijn gecombineerd, moet deze tekenreeks verder worden beveiligd door er het hashinglogaritme SHA-256 op toe te passen. Door de waarde te hashen wordt het voor een aanvaller moeilijker deze te reverse-engineeren.

Op dit moment heeft de code een ByteArray met de naam `unhashedKey` die het samengevoegde wachtwoord bevat dat met de salt is gecombineerd. Het as3corelib-project (de kernbibliotheek van ActionScript 3.0) bevat een klasse SHA256 in het pakket com.adobe.crypto. De methode `SHA256.hashBytes()` die een SHA-256-hash uitvoert voor een ByteArray en een tekenreeks retourneert die het hashresultaat van 256 bits bevat als een hexadecimaal getal. De klasse EncryptionKeyGenerator gebruikt de klasse SHA256 om de sleutel te husselen (hashing):

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

De coderingssleutel uit de hash extraheren

Adobe AIR 1.5 of hoger

De coderingssleutel moet een ByteArray zijn die precies 16 bytes (128 bits) lang is. Het resultaat van het hashalgoritme SHA-256 is altijd 256 bits lang. Dat betekent dat in de laatste stap 128 bits uit het hashresultaat gehaald moeten worden die de uiteindelijke coderingssleutel gaan vormen.

In de klasse EncryptionKeyGenerator reduceert de code de sleutel tot 128 bits door de methode `generateEncryptionKey()` aan te roepen. Vervolgens wordt het resultaat van die methode geretourneerd als het resultaat van de methode `getEncryptionKey()`:

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

Het is niet noodzakelijk om de eerste 128 bits als coderingssleutel te gebruiken. U zou een reeks bits kunnen selecteren vanaf een willekeurig punt, u zou elke andere bit kunnen selecteren, of bits op een andere manier kunnen selecteren. Essentieel is dat de code 128 specifieke bits selecteert en dat diezelfde 128 bits iedere keer opnieuw worden gebruikt.

In dit geval gebruikt de methode `generateEncryptionKey()` de reeks bits die op de achttiende byte begint, als coderingssleutel. Zoals eerder vermeld geeft de klasse SHA256 als resultaat een tekenreeks met een 256-bits hash als hexadecimaal getal. Eén enkel blok van 128 bits heeft te veel bytes om in één keer aan een ByteArray toe te voegen. De code gebruikte daarom een `for`-lus om tekens uit de hexadecimale tekenreeks te extraheren, deze te converteren naar numerieke waarden en die toe te voegen aan de ByteArray. De resultaatreeks van SHA-256 is 64 tekens lang. Een bereik van 128 bits is gelijk aan 32 tekens in de tekenreeks, en elk teken vertegenwoordigt 4 bits. De kleinste gegevenstoename die u kunt toevoegen aan een ByteArray is 1 byte (8 bits), hetgeen equivalent is aan twee tekens in de `hash`-tekenreeks. De lus telt daarom van 0 tot 31 (32 tekens) in stappen van 2 tekens.

Binnen de lus bepaalt de code eerst de beginpositie voor het huidige paar tekens. Aangezien het gewenste bereik start bij het teken op indexpositie 17 (de achttiende byte), wordt aan de variabele `position` de huidige iteratorwaarde (`i`) plus 17 toegekend. De code gebruikt de methode `substr()` van het tekenreeksobject om de twee tekens op de huidige positie te extraheren. Deze twee tekens worden opgeslagen in de variabele `hex`. Vervolgens gebruikt de code de methode `parseInt()` om de `hex`-tekenreeks te converteren naar een decimaal geheel getal. Deze slaat die waarde op in de int-variabele `byte`. Tot slot voegt de code de waarde in `byte` toe aan de ByteArray `result` met behulp van de methode `writeByte()`. Wanneer de lus gereed is, bevat de ByteArray `result` 16 bytes en is deze gereed voor gebruik als databasecoderingssleutel.

```
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;
}
```

Strategieën voor het werken met SQL-databases

Adobe AIR 1.0 of hoger

Er zijn verschillende manieren waarop een toepassing een lokale SQL-database kan benaderen en ermee kan werken. Het ontwerp van de toepassing kan variëren op het gebied van de structuur van de toepassingscode, de sequentie en timing van de uitvoering van de bewerkingen enzovoort. De technieken die u kiest, kunnen een invloed hebben op het gemak waarmee uw toepassing kan worden ontwikkeld. Ze kunnen een invloed hebben op het gemak waarmee de toepassing in de toekomst kan worden aangepast. Ze kunnen ook een invloed hebben op de prestaties van de toepassing voor de gebruiker.

Niet-lege databases distribueren

Adobe AIR 1.0 of hoger

Als u een lokale SQL-database van AIR in uw toepassing gebruikt, verwacht de toepassing een database met een bepaalde structuur (tabellen, kolommen enzovoort). Sommige toepassingen verwachten ook bepaalde gegevens in het databasebestand. U kunt op verschillende manieren zorgen dat de database de juiste structuur heeft, onder andere door de database binnen de toepassingscode te creëren. Wanneer de toepassing wordt geladen, controleert deze of het gekoppelde databasebestand zich op een bepaalde locatie bevindt. Als dat niet het geval is, voert de toepassing een reeks opdrachten uit om het databasebestand te creëren, de databasestructuur toe te passen en de tabellen te vullen met de initiële gegevens.

De code die de database en de overeenkomstige tabellen creëert, is vaak complex. Hoewel deze code doorgaans slechts één keer tijdens de levensduur van de geïnstalleerde toepassing wordt gebruikt, wordt de toepassing er groter en complexer door. Een alternatief voor het programmamatig creëren van de database, structuur en gegevens is uw toepassing te distribueren met een niet-lege database. Hiervoor neemt u het databasebestand op in het AIR-pakket van de toepassing.

Net als alle bestanden die in een AIR-pakket worden opgenomen, wordt een gebundeld databasebestand geïnstalleerd in de toepassingsmap (de map die wordt ingesteld door de eigenschap `File.applicationDirectory`). Bestanden in die map zijn echter alleen-lezen. Gebruik het bestand uit het AIR-pakket als "sjabloon"database. De eerste keer dat een gebruiker de toepassing uitvoert, zorgt u dat het oorspronkelijke databasebestand naar de "De opslagmap van een toepassing aanwijzen" op pagina 710 van de gebruiker (of een andere locatie) wordt gekopieerd. Vervolgens wordt die database in de toepassing gebruikt.

Aanbevolen procedures voor het werken met lokale SQL-databases

Adobe AIR 1.0 of hoger

Hieronder volgt een lijst van suggesties voor technieken die u kunt toepassen om de prestaties, de beveiliging en het onderhoudsgemak van uw toepassingen te verhogen bij het werken met lokale SQL-databases.

Creëer vooraf databaseverbindingen

Adobe AIR 1.0 of hoger

Zelfs als uw toepassing geen instructies uitvoert wanneer deze wordt geladen, maakt u een instantie van een SQLConnection-object en roept u de methode `open()` of `openAsync()` van de instantie zo vroeg mogelijk op (bijvoorbeeld onmiddellijk nadat de toepassing is gestart) om vertraging bij het uitvoeren van instructies te voorkomen. Zie "Verbinden met een database" op pagina 767.

Gebruik databaseverbindingen opnieuw

Adobe AIR 1.0 of hoger

Als u tijdens de uitvoering van uw toepassing een bepaalde database benadert, zorgt u voor een verwijzing naar de SQLConnection-instantie en gebruikt u deze altijd opnieuw in de hele toepassing zodat u de verbinding niet elke keer hoeft te sluiten en weer te openen. Zie "Verbinden met een database" op pagina 767.

Gebruik bij voorkeur de asynchrone uitvoeringsmodus

Adobe AIR 1.0 of hoger

Bij het schrijven van code voor het benaderen van gegevens kan het verleidelijk zijn om bewerkingen synchroon in plaats van asynchroon te laten uitvoeren omdat het gebruik van synchrone bewerkingen vaak kortere en minder complexe code vereist. Zoals is beschreven in "Synchrone en asynchrone databasebewerkingen gebruiken" op pagina 795 kunnen synchrone bewerkingen de prestaties echter zo verminderen dat de gebruiker dit merkt, zodat het werken met uw toepassing minder aantrekkelijk wordt. De tijd die nodig is om één bewerking uit te voeren, is afhankelijk van de bewerking en met name van de hoeveelheid gegevens die moet worden verwerkt. Bijvoorbeeld: een SQL-instructie van het type `INSERT` die maar één rij aan de database toevoegt, heeft minder tijd nodig dan een

instructie van het type `SELECT` die duizenden rijen gegevens moet ophalen. Wanneer u echter de synchrone modus gebruikt om meerdere bewerkingen uit te voeren, worden de bewerkingen doorgaans samengeperst. Zelfs als elke bewerking slechts een korte uitvoeringstijd heeft, blokkeert de toepassing tot alle synchrone bewerkingen zijn voltooid. Hierdoor is de totale uitvoeringstijd van meerdere samengeperste bewerkingen mogelijk lang genoeg om uw toepassing onbruikbaar te maken.

Gebruik daarom standaard de asynchrone modus, met name voor bewerkingen met een groot aantal rijen. Er bestaat een techniek om de verwerking van grote resultaatsets van de instructie `SELECT` te verdelen. Zie hiervoor "SELECT-resultaten in delen ophalen" op pagina 783. Deze techniek kan echter alleen in de asynchrone uitvoeringsmodus worden gebruikt. Gebruik alleen synchrone bewerkingen wanneer u een bepaald resultaat niet met asynchrone programmering kunt verkrijgen, wanneer u de afname aan prestaties waarmee de gebruikers van de toepassing worden geconfronteerd in overweging hebt genomen, en wanneer u de toepassing hebt getest zodat u weet wat de invloed op de prestaties van de toepassing zal zijn. Het gebruik van de asynchrone uitvoeringsmodus kan het coderen van de toepassing complexer maken. Vergeet echter niet dat u de code maar één hoeft te schrijven maar de gebruikers van uw toepassing deze vele keren moeten gebruiken, traag of snel.

In vele gevallen kan een groot aantal SQL-bewerkingen tegelijk in de wachtrij worden geplaatst wanneer u een aparte SQLStatement-instantie voor elke uit te voeren SQL-instructie gebruikt, waardoor asynchrone code op ongeveer dezelfde manier als synchrone code wordt geschreven. Zie "De asynchrone uitvoeringsmodus" op pagina 798 voor meer informatie.

Gebruik aparte SQL-instructies en wijzig de eigenschap 'text' van de SQLStatement niet

Adobe AIR 1.0 of hoger

Creëer voor elke SQL-instructie die meer dan één keer in de toepassing wordt uitgevoerd, een aparte SQLStatement-instantie voor elke SQL-instructie. Gebruik die SQLStatement-instantie elke keer dat de overeenkomstige SQL-opdracht wordt uitgevoerd. Als u bijvoorbeeld een toepassing schrijft met vier verschillende SQL-bewerkingen die meerdere keren worden uitgevoerd. In dat geval creëert u vier aparte SQLStatement-instanties en roept u de methode `execute()` van elke instructie op om deze uit te voeren. U kunt ook één SQLStatement-instantie voor alle SQL-instructies gebruiken, waarbij de eigenschap `text` van de instantie elke keer opnieuw wordt gedefinieerd voordat de instructie wordt uitgevoerd. Deze techniek wordt echter afgeraden.

Gebruik instructieparameters

Adobe AIR 1.0 of hoger

Gebruik SQLStatement-parameters en voeg gebruikersinvoer niet samen tot instructietekst. Het gebruik van parameters maakt uw toepassing veiliger omdat het SQL-injectieaanvallen onmogelijk maakt. Hierdoor kunt u objecten gebruiken in query's (in plaats van alleen literale SQL-waarden). Bovendien worden de instructies efficiënter uitgevoerd omdat ze opnieuw kunnen worden gebruikt zonder dat ze opnieuw hoeven te worden gecompileerd elke keer dat ze worden uitgevoerd. Zie "Parameters gebruiken in instructies" op pagina 771 voor meer informatie.

Hoofdstuk 41: Werken met bytearrays

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse ByteArray kunt u een binaire gegevensstroom, die in feite een array met bytes is, lezen en schrijven. Met deze klasse krijgt u toegang tot gegevens op het meest elementaire niveau. Omdat computergegevens bestaan uit bytes, of groepen van 8 bits, kunt u door het lezen van gegevens in bytes toegang krijgen tot gegevens waarvoor geen klassen en toegangsmethoden bestaan. Met de klasse ByteArray kunt u elke gegevensstroom, van een bitmap tot een gegevensstroom die via het netwerk wordt verzonden, parseren op byteniveau.

Met de methode `writeObject()` kunt u een object in geserialiseerde AMF-indeling (Action Message Format) schrijven naar een bytearray, terwijl u met de methode `readObject()` een geserialiseerd object uit een bytearray kunt inlezen in een variabele van het oorspronkelijke gegevenstype. U kunt alle objecten serialiseren, behalve weergaveobjecten (objecten die op de weergavelijst kunnen worden geplaatst). U kunt geserialiseerde objecten ook opnieuw aan aangepaste klasse-instanties toewijzen als de aangepaste klasse beschikbaar is voor de runtime. Nadat een object naar AMF is geconverteerd, kunt u het verzenden via een netwerkverbinding of opslaan in een bestand.

De voorbeeldtoepassing Adobe® AIR® die hier wordt beschreven leest een .zip-bestand als een voorbeeld van het verwerken van een bytestroom, het extraheren van een lijst bestanden die het .zip-bestand bevat en deze naar het bureaublad schrijft.

Meer Help-onderwerpen

flash.utils.ByteArray

flash.utils.IExternalizable

AMF-specificatie (Action Message Format)

Bytearrays lezen en schrijven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse ByteArray is een onderdeel van het pakket flash.utils. Als u een ByteArray-object wilt maken in ActionScript 3.0, importeert u de klasse ByteArray en roept u de constructor op, zoals weergegeven in het volgende voorbeeld:

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

ByteArray, methoden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Elke zinvolle gegevensstroom wordt geordend in een indeling waarin u de gewenste informatie kunt zoeken. Een record in een eenvoudig werknemersbestand kan bijvoorbeeld een id-nummer, een naam, een adres, een telefoonnummer enzovoort bevatten. Een MP3-geluidsbestand bevat een ID3-code met de titel, de auteur, het album, de publicatiedatum en het genre van het bestand dat wordt gedownload. Door deze indeling weet u in welke volgorde de gegevens in de gegevensstroom zijn opgenomen. U kunt de bytestroom daardoor op een intelligente manier lezen.

De klasse ByteArray bevat diverse methoden waarmee u een gegevensstroom eenvoudiger kunt lezen of schrijven. Enkele van deze methoden zijn: `readBytes()` en `writeBytes()`, `readInt()` en `writeInt()`, `readFloat()` en `writeFloat()`, `readObject()` en `writeObject()`, en `readUTFBytes()` en `writeUTFBytes()`. Met deze methoden kunt u gegevens uit een gegevensstroom inlezen in variabelen van bepaalde gegevenstypen en vanuit bepaalde gegevenstypen rechtstreeks naar de binaire gegevensstroom schrijven.

Met de volgende code wordt bijvoorbeeld een eenvoudige array met tekenreeksen en getallen met een drijvende komma gelezen en wordt elk element naar een bytearray geschreven. Door de indeling van de array kan de code de juiste ByteArray-methoden (`writeUTFBytes()` en `writeFloat()`) oproepen voor het schrijven van de gegevens. Door het herhaalde gegevenspatroon wordt het mogelijk de array in een lus te lezen.

```
 // 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
```

De eigenschap position

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de eigenschap position wordt de huidige positie opgeslagen van de pointer die de index van de bytearray bepaalt tijdens het lezen of schrijven. De initiële waarde van de eigenschap position is 0 (nul), zoals wordt weergegeven in de volgende code:

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

Wanneer u een bytearray leest of er naar schrijft, wordt de eigenschap position bijgewerkt met de methode die u gebruikt, zodat wordt verwezen naar de locatie die volgt op de laatste byte die is gelezen of geschreven. Met de volgende code wordt bijvoorbeeld een tekenreeks naar een bytearray geschreven en verwijst de eigenschap position daarna naar de byte die volgt op de tekenreeks in de 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
```

Bij een leesbewerking wordt de eigenschap position verhoogd met het aantal gelezen bytes.

```
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!
```

U kunt de eigenschap position instellen op een specifieke locatie in de bytearray om vanaf dat punt te lezen of te schrijven.

De eigenschappen bytesAvailable en length

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschappen `length` en `bytesAvailable` geven aan hoe lang een bytearray is en hoeveel bytes er aanwezig zijn vanaf de huidige positie tot het einde. In het volgende voorbeeld ziet u hoe u deze eigenschappen kunt gebruiken. In het voorbeeld wordt een tekenreeks met tekst naar de bytearray geschreven en worden vervolgens alle bytes in de array een voor een gelezen totdat het teken 'a' of het einde (`bytesAvailable <= 0`) wordt aangetroffen.

```
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
}
```

De eigenschap endian

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Getallen die uit meerdere bytes bestaan (getallen die in meer dan 1 byte geheugen worden opgeslagen), kunnen op verschillende manieren op computers worden opgeslagen. Een geheel getal kan bijvoorbeeld 4 bytes (32 bits) geheugen in beslag nemen. Op sommige computers wordt de meest significante byte van het getal eerst opgeslagen, op het laagste geheugenadres, en op andere wordt de minst significante byte eerst opgeslagen. Dit kenmerk van een computer (de bytevolgorde) wordt *big endian* (meest significante byte eerst) of *little endian* (minst significante byte eerst) genoemd. Het getal 0x31323334 wordt bijvoorbeeld als volgt opgeslagen bij de bytevolgorde big endian en little endian, waarbij a0 het laagste geheugenadres van de 4 bytes is en a3 het hoogste:

Big Endian	**Big Endian**	**Big Endian**	**Big Endian**
a0	a1	a2	a3
31	32	33	34

Little Endian	Little Endian	Little Endian	Little Endian
a0	a1	a2	a3
34	33	32	31

Met de eigenschap `endian` van de klasse ByteArray kunt u deze bytevolgorde aangeven voor getallen die uit meerdere bytes bestaan. De geaccepteerde waarden voor deze eigenschap zijn `"bigEndian"` en `"littleEndian"` en in de klasse Endian zijn de constanten `BIG_ENDIAN` en `LITTLE_ENDIAN` gedefinieerd om deze tekenreeksen in te stellen voor de eigenschap `endian`.

De methoden compress() en uncompress()

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `compress()` kunt u een bytearray comprimeren volgens een compressiealgoritme die u opgeeft als een parameter. Met de methode `uncompress()` kunt u een bytearray decomprimeren volgens een compressiealgoritme. Nadat u `compress()` en `uncompress()` hebt opgeroepen, wordt de lengte van de bytearray ingesteld op de nieuwe lengte en wordt de eigenschap position ingesteld op het einde.

In de klasse CompressionAlgorithm zijn constanten gedefinieerd waarmee u de compressiealgoritme kunt opgeven. De klasse ByteArray biedt ondersteuning voor de volgende algoritmen: Deflate (alleen AIR), ZLIB en LZMA. De ZLIB-gecomprimeerde gegevensindeling wordt beschreven in http://www.ietf.org/rfc/rfc1950.txt. De lzma-algoritme is toegevoegd in Flash Player 11.4 en AIR 3.4 en wordt beschreven in http://www.7-zip.org/7z.html.

De Deflate-compressiealgoritme wordt gebruikt in diverse compressie-indelingen, zoals ZLIB, GZIP en enkele ZIP-implementaties. De Deflate-compressiealgoritme wordt beschreven in http://www.ietf.org/rfc/rfc1951.txt.

In het volgende voorbeeld wordt een bytearray met de naam `bytes` gecomprimeerd met de Izma-algoritme:

```
bytes.compress(CompressionAlgorithm.LZMA);
```

In het volgende voorbeeld wordt een gecomprimeerde bytearray gedecomprimeerd met de Deflate-algoritme:

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

Objecten lezen en schrijven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methoden `readObject()` en `writeObject()` leest u een object in een bytearray en schrijft u een object naar een bytearray, gecodeerd in geserialiseerde AMF-indeling (Action Message Format). AMF is een berichtenprotocol dat door Adobe is gemaakt en wordt gebruikt door diverse ActionScript 3.0-klassen, zoals Netstream, NetConnection, NetStream, LocalConnection en Shared Objects.

Een typemarkering van één byte bevat de beschrijving van het type van de gecodeerde gegevens die volgen. AMF gebruikt de volgende 13 gegevenstypen:

```
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
```

De gecodeerde gegevens volgen op de typemarkering, tenzij de markering één mogelijke waarde bevat, zoals null of true of false, zodat verder niets wordt gecodeerd.

Er zijn twee versies van AMF: AMF0 en AMF3. AMF 0 ondersteunt het verzenden van complexe objecten via verwijzingen en staat eindpunten toe voor het herstellen van objectrelaties. AMF 3 is verbeterd ten opzichte van AMF 0 doordat naast objectverwijzingen ook objectkenmerken en tekenreeksen kunnen worden verzonden en nieuwe gegevenstypen worden ondersteund die in ActionScript 3.0 zijn geïntroduceerd. De eigenschap `ByteArray.objectEcoding` geeft aan welke versie van AMF wordt gebruikt om de objectgegevens te coderen. In de klasse flash.net.ObjectEncoding zijn constanten gedefinieerd waarmee de AMF-versie wordt opgegeven: `ObjectEncoding.AMF0` en `ObjectEncoding.AMF3`.

In het volgende voorbeeld wordt `writeObject()` opgeroepen om een XML-object te schrijven naar een bytearray die daarna met de Deflate-algoritme wordt gecomprimeerd en in het bestand `order` op de desktop wordt geschreven. In dit voorbeeld wordt een label gebruikt om het bericht "Wrote order file to desktop!" in het AIR-venster weer te geven nadat de bewerking is voltooid.

```
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();
    }
}
```

Met de methode `readObject()` wordt een object in geserialiseerde AMF-indeling ingelezen vanuit een bytearray en opgeslagen in een object van het opgegeven type. In het volgende voorbeeld wordt het bestand `order` vanaf de desktop ingelezen in een bytearray (`inBytes`). Vervolgens wordt het bestand gedecomprimeerd en wordt `readObject()` opgeroepen om het op te slaan in het XML-object `orderXML`. In het voorbeeld wordt elk knooppunt in een lusconstructie `for each()` toegevoegd aan een tekstgebied voor weergave. In het voorbeeld wordt ook de waarde van de eigenschap `objectEncoding` weergegeven, evenals een header voor de inhoud van het bestand `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();
    }
}
```

Voorbeeld van een bytearray: een ZIP-bestand lezen

Adobe AIR 1.0 of hoger

In dit voorbeeld ziet u hoe een eenvoudig ZIP-bestand wordt gelezen dat verschillende typen bestanden bevat. Hiervoor worden relevante gegevens uit de metagegevens van elk bestand opgehaald, waarbij elk bestand in een bytearray wordt gedecomprimeerd en naar de desktop wordt geschreven.

De algemene structuur van een ZIP-bestand is gebaseerd op de specificatie van PKWARE Inc., die u kunt vinden op http://www.pkware.com/documents/casestudies/APPNOTE.TXT. Eerst staan er een bestandsheader en bestandsgegevens voor het eerste bestand in het ZIP-bestand, gevolgd door een combinatie van een bestandsheader en bestandsgegevens voor elk ander bestand. (De structuur van de bestandsheader wordt later beschreven.) Daarna bevat het ZIP-bestand eventueel een record met een gegevensdescriptor (gewoonlijk wanneer het ZIP-bestand in het geheugen is gemaakt in plaats van opgeslagen op een schijf). Hierna volgen diverse optionele elementen: decoderingsheader van archief, extra gegevensrecord voor archief, centrale mapstructuur, Zip64-einde van centrale maprecord, Zip64-einde van centrale maplocator en het einde van de centrale maprecord.

De code in dit voorbeeld is alleen geschreven om ZIP-bestanden te parseren die geen mappen bevatten en er worden geen records met een gegevensdescriptor verwacht. Alle informatie na de laatste bestandsgegevens wordt genegeerd.

De bestandsheader voor elk bestand heeft de volgende indeling:

handtekening van bestandsheader	4 bytes
vereiste versie	2 bytes
algemene bitmarkering	2 bytes
compressiemethode	2 bytes (8=DEFLATE; 0=UNCOMPRESSED)
tijd van laatste bestandswijziging	2 bytes
datum van laatste bestandswijziging	2 bytes

crc-32	4 bytes
gecomprimeerde grootte	4 bytes
ongecomprimeerde grootte	4 bytes
lengte van bestandsnaam	2 bytes
lengte van extra veld	2 bytes
bestandsnaam	variabele
extra veld	variabele

Na de bestandsheader volgen de werkelijke bestandsgegevens die gecomprimeerd of niet kunnen zijn, afhankelijk van de markering voor de compressiemethode. De markering is 0 (nul) als de bestandsgegevens niet zijn gecomprimeerd, 8 als de gegevens zijn gecomprimeerd met de Deflate-algoritme of een andere waarde voor andere compressiealgoritmen.

De gebruikersinterface voor dit voorbeeld bestaat uit een label en een tekstgebied (`taFiles`). De toepassing schrijft de volgende informatie naar het tekstgebied voor elk bestand dat in het ZIP-bestand wordt aangetroffen: bestandsnaam, gecomprimeerde grootte en niet-gecomprimeerde grootte. Het volgende MXML-document bepaalt de gebruikersinterface voor de Flex-versie van de toepassing:

```
<?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>
```

Aan het begin van het programma worden de volgende taken uitgevoerd:

- De vereiste klassen worden geïmporteerd.

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- De gebruikersinterface wordt gedefinieerd. voor 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);
```

- De bytearray `bytes` wordt gedefinieerd.

```
var bytes:ByteArray = new ByteArray();
```

- Er worden variabelen gedefinieerd voor het opslaan van metagegevens uit de bestandsheader.

```
// 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;
```

- De objecten File (`zfile`) en FileStream (`zStream`) worden gedefinieerd voor het ZIP-bestand en de locatie wordt opgegeven van het ZIP-bestand waaruit de bestanden worden opgehaald: een bestand met de naam "HelloAIR.zip" in de desktopmap.

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

In Flex start de programmacode in de `init()`-methode, die wordt opgeroepen als de `creationComplete`-handler voor de root van de `mx:WindowedApplication`-tag.

```
// for Flex
private function init():void
{
```

Het programma begint met het openen van het ZIP-bestand in de leesmodus.

```
zStream.open(zfile, FileMode.READ);
```

Vervolgens wordt de eigenschap `endian` van `bytes` op `LITTLE_ENDIAN` ingesteld om aan te geven dat de minst significante byte eerst wordt aangegeven in de bytevolgorde van numerieke velden.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

Vervolgens wordt met de instructie `while()` een lus gestart die wordt uitgevoerd tot de huidige positie in de bestandsstroom groter is dan of gelijk is aan de grootte van het bestand.

```
while (zStream.position < zfile.size)
{
```

Met de eerste instructie in de lus worden de eerste 30 bytes van de bestandsstroom ingelezen in de bytearray `bytes`. De eerste 30 bytes bevatten het gedeelte van de eerste bestandsheader dat een vaste grootte heeft.

```
// read fixed metadata portion of local file header
zStream.readBytes(bytes, 0, 30);
```

Vervolgens wordt een geheel getal (`signature`) gelezen uit de eerste bytes van de header die uit 30 bytes bestaat. In de definitie van de ZIP-indeling is opgegeven dat de handtekening voor elke bestandsheader de hexadecimale waarde `0x04034b50` heeft; als de handtekening anders is, betekent dit dat het einde van het bestandsgedeelte van het ZIP-bestand is bereikt en dat er geen bestanden meer zijn om uit te pakken. In dat geval wordt de `while`-lus onmiddellijk beëindigd en wordt niet gewacht tot het einde van de bytearray is bereikt.

```
bytes.position = 0;
signature = bytes.readInt();
// if no longer reading data files, quit
if (signature != 0x04034b50)
{
    break;
}
```

In het volgende deel van de code wordt de headerbyte op positie 8 gelezen en wordt de waarde opgeslagen in de variabele `compMethod`. Deze byte bevat een waarde die aangeeft met welke compressiemethode dit bestand is gecomprimeerd. Er zijn diverse compressiemethoden toegestaan, maar in de praktijk gebruiken bijna alle ZIP-bestanden de Deflate-compressiealgoritme. Als het huidige bestand is gecomprimeerd met Deflate-compressie, `is compMethod` 8; als het bestand niet is gecomprimeerd, is `compMethod` 0.

```
bytes.position = 8;
compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

Na de eerste 30 bytes volgt een gedeelte van de header met een variabele lengte dat de bestandsnaam en eventueel een extra veld bevat. In de variabele `offset` is de grootte van dit gedeelte opgeslagen. De grootte wordt berekend door de lengte van de bestandsnaam en de lengte van het extra veld op te tellen. Deze waarden worden ingelezen uit de header op positie 26 en 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
```

Vervolgens wordt uit het gedeelte van de bestandsheader met variabele lengte het aantal bytes ingelezen dat is opgeslagen in de variabele `offset`.

```
// read variable length bytes between fixed-length header and compressed file data
zStream.readBytes(bytes, 30, offset);
```

De bestandsnaam wordt ingelezen vanuit het gedeelte van de header met variabele lengte en samen met de gecomprimeerde en niet-gecomprimeerde (oorspronkelijke) grootte van het bestand weergegeven in het tekstgebied.

```
// 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';
```

In het voorbeeld wordt de rest van het bestand vanuit de bestandsstroom ingelezen in `bytes` voor de lengte die is opgegeven door de gecomprimeerde grootte, waarbij de bestandsheader in de eerste 30 bytes wordt overschreven. De gecomprimeerde grootte is nauwkeurig, zelfs als het bestand niet is gecomprimeerd, omdat de gecomprimeerde grootte in dat geval gelijk is aan de niet-gecomprimeerde grootte van het bestand.

```
    // 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);
```

Vervolgens wordt het gecomprimeerde bestand gedecomprimeerd en wordt de functie `outfile()` opgeroepen om het naar de uitvoerstroom te schrijven. De bestandsnaam en de bytearray die de bestandsgegevens bevat, worden doorgegeven aan `outfile()`.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

In het eerder gegeven voorbeeld werkt `bytes.uncompress(CompressionAlgorithm.DEFLATE)` alleen in AIR-toepassingen. Als u verkleinde gegevens wilt decomprimeren voor zowel AIR als Flash Player, roept u de ByteArray-functie `inflate()` op.

De accolades sluiten geven het einde van de `while`-lus aan en van de `init()`-methode en de Flex-toepassingscode, behalve voor de `outFile()`-methode. De `while`-lus wordt opnieuw gestart en de volgende bytes in het ZIP-bestand worden verwerkt: er wordt een ander bestand uitgepakt of de verwerking van het ZIP-bestand wordt beëindigd als het laatste bestand is verwerkt.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

Met de functie `outfile()` wordt in de schrijfmodus op de desktop een uitvoerbestand geopend dat de naam krijgt die is opgegeven met de parameter`filename`. Vervolgens wordt het gegevensbestand vanuit de parameter `data` naar de uitvoerstroom (`outStream`) geschreven en wordt het bestand gesloten.

```
// 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();
}
```

Hoofdstuk 42: Basisinformatie over netwerken en communicatie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u toepassingen maakt in Flash Player of AIR, hebt u vaak toegang nodig tot resources buiten uw toepassing. U wilt bijvoorbeeld een aanvraag voor een afbeelding indienen bij een internetwebserver, zodat u de afbeeldingsgegevens ontvangt. Of u wilt misschien objecten met serienummering heen en weer sturen over een socketverbinding met een toepassingsserver. Met de API's van Flash Player en AIR beschikt u over verschillende klassen om uw toepassing te laten deelnemen aan deze gegevensuitwisseling. Deze API's bieden IP-netwerkondersteuning voor verschillende protocollen, zoals UDP, TCP, HTTP, RTMP en RTMFP.

Met de volgende klassen kunt u gegevens over een netwerk verzenden en ontvangen:

Klasse	Ondersteunde gegevensopmaak	Protocollen	Beschrijving
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	Hiermee worden ondersteunde gegevenstypen geladen, waarna de gegevens worden geconverteerd naar een weergaveobject. Zie "Scherminhoud dynamisch laden" op pagina 210.
URLLoader	Alle (tekst, XML, binair, enzovoort.)	HTTP, HTTPS	Hiermee worden gegevens met een willekeurige opmaak geladen. Uw toepassing is verantwoordelijk voor de gegevensinterpretatie. Zie "De klasse URLLoader gebruiken" op pagina 862
FileReference	Alle	HTTP	Bestanden uploaden en downloaden. Zie "De FileReference-klasse gebruiken" op pagina 689
NetConnection	Video, audio, AMF (ActionScript Message Format)	HTTP, HTTPS, RTMP, RTMFP	Hiermee wordt verbinding gemaakt met video, audio en externe objectstreams. Zie "Werken met video" op pagina 499.
Sound	Audio	HTTP	Hiermee worden ondersteunde audio-indelingen geladen en afgespeeld. Zie "Externe geluidsbestanden laden" op pagina 467.
XMLSocket	XML	TCP	Hiermee worden XML-berichten uitgewisseld met een XMLSocket-server. Zie "XML-sockets" op pagina 848.
Socket	Alle	TCP	Hiermee wordt een verbinding gemaakt met een TCP-socketserver Zie "Binaire clientsockets" op pagina 843.
SecureSocket (AIR)	Alle	TCP met SSLv3 of TLSv1	Hiermee wordt een verbinding gemaakt met een TCP-socketserver waarvoor SSL- of TLS-beveiliging is vereist. Zie "Beveiligde clientsockets (AIR)" op pagina 843.

Klasse	Ondersteunde gegevensopmaak	Protocollen	Beschrijving
ServerSocket (AIR)	Alle	TCP	Handelt als een server voor binnenkomende TCP-socketverbindingen. Zie "Serversockets" op pagina 852.
DatagramSocket (AIR)	Alle	UDP	Verzendt en ontvangt UDP-pakketten. Zie "UDP-sockets (AIR)" op pagina 854

Bij het maken van een webtoepassing is het vaak handig als permanente informatie over de status van de gebruikerstoepassing kan worden opgeslagen. HTML-pagina's en toepassingen gebruiken hiervoor cookies. In Flash Player kunt u hiervoor de SharedObject-klasse gebruiken. Zie "Gezamenlijke objecten" op pagina 741. (In AIR-toepassingen kunt u ook de SharedObject-klasse gebruiken, maar als u de gegevens opslaat naar een gewoon bestand zijn er minder beperkingen.)

Wanneer uw Flash Player- of AIR-toepassing moet communiceren met een andere Flash Player- of AIR-toepassing op dezelfde computer, kunt u de LocalConnection-klasse gebruiken. Zo kunnen twee (of meer) SWF-bestanden op dezelfde webpagina met elkaar communiceren. Evenzo geldt dat een SWF-bestand dat op een webpagina wordt uitgevoerd, kan communiceren met een AIR-toepassing. Zie "Communiceren met andere instanties van Flash Player en AIR" op pagina 877.

Wanneer u moet communiceren met andere, niet-SWF-processen op de lokale computer, kunt u de NativeProcess-klasse gebruiken, die is toegevoegd in AIR 2. Met de NativeProcess-klasse kan uw AIR-toepassing communiceren met andere toepassingen en ook dergelijke toepassingen starten. Zie "Communiceren met native processen in AIR" op pagina 884.

Wanneer u informatie wilt ophalen over de netwerkomgeving van de computer waarop de AIR-toepassing wordt uitgevoerd, kunt u de volgende klassen gebruiken:

- NetworkInfo: deze klasse biedt informatie over de beschikbare netwerkinterfaces, zoals het IP-adres van de computer. Zie "Netwerkinterfaces" op pagina 835.
- DNSResolver: hiermee kunt u DNS-records opzoeken. Zie "DNS-records (Domain Name System)" op pagina 839.
- ServiceMonitor: hiermee kunt u de beschikbaarheid van een server nagaan. Zie "Servicebewaking" op pagina 837.
- URLMonitor: hiermee kunt u de beschikbaarheid van een resource bij een bepaalde URL controleren. Zie "HTTP-bewaking" op pagina 838.
- SocketMonitor en SecureSocketMonitor: hiermee kunt u de beschikbaarheid van een resource bij een socket controleren. Zie "Socketbewaking" op pagina 839.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot het programmeren van netwerk- en communicatiecodes.

Externe gegevens Gegevens die in een of andere vorm buiten de AIR-toepassing zijn opgeslagen en die wanneer nodig in de toepassing worden geladen. Deze gegevens kunnen zijn opgeslagen in een bestand dat rechtstreeks wordt geladen maar ook in een database of op andere wijze, waarbij ze worden opgehaald door scripts die worden aangeroepen of programma's die op een server worden uitgevoerd.

URL-gecodeerde variabelen De URL-gecodeerde indeling biedt een manier om meerdere variabelen (paren van namen en waarden van variabelen) in één tekenreeks weer te geven. Afzonderlijke variabelen worden geschreven in de indeling naam=waarde. Elke waarde (dat wil zeggen elk naam-waardepaar) wordt van de andere waarden gescheiden door het en-teken (&), zoals in het volgende voorbeeld wordt geïllustreerd: variabele1=waarde1&variabele2=waarde2. Zo kan een onbeperkt aantal variabelen worden verzonden als één bericht.

MIME-type Een standaardcode die wordt gebruikt om het type van een bepaald bestand tijdens internetcommunicatie aan te duiden. Elk bestandstype heeft een specifieke code waarmee dit type wordt aangeduid. Wanneer een bestand of bericht wordt verzonden, geeft de computer (zoals een webserver of de Flash Player- of AIR-instantie van een gebruiker) het type op van het bestand dat wordt verzonden.

HTTP Hypertext Transfer Protocol - een standaardindeling voor het aanbieden van webpagina's en diverse andere soorten inhoud die via internet worden verzonden.

Aanvraagmethode Wanneer een toepassing (zoals een AIR-toepassing of een webbrowser) een bericht (een zogenaamde HTTP-aanvraag) naar een webserver verzendt, kunnen eventuele gegevens die worden verzonden, aan de hand van twee methoden in de aanvraag worden ingesloten. Die twee aanvraagmethoden zijn GET en POST. Het programma op de server dat de aanvraag ontvangt, moet in het juiste gedeelte van de aanvraag kijken om de gegevens te vinden. Daarom moet de aanvraagmethode die wordt gebruikt om gegevens te verzenden vanuit uw toepassing, overeenkomen met de aanvraagmethode die wordt gebruikt om die gegevens op de server te lezen.

Socketverbinding Een blijvende verbinding voor communicatie tussen twee computers.

Uploaden Een bestand naar een andere computer verzenden.

Downloaden Een bestand van een andere computer ophalen.

Netwerkinterfaces

Adobe AIR 2 of hoger

U kunt het object NetworkInfo gebruiken om te zien welke hardware- en softwarenetwerkinterfaces beschikbaar zijn voor uw toepassing. Het object NetworkInfo is een *singleton*-object. U hoeft dit object niet te maken. Gebruik in plaats hiervan de statische klasseneigenschap `networkInfo` om toegang te krijgen tot het enkele object NetworkInfo. Het object NetworkInfo verzendt eveneens een `networkChange`-gebeurtenis als er zich een wijziging voordoet in een van de beschikbare interfaces.

Roep de methode `findInterfaces()` aan om een lijst met NetworkInterface-objecten te zien. Elk NetworkInterface-object in de lijst geeft een beschrijving van een van de beschikbare interfaces. Het NetworkInterface-object geeft informatie over onder meer het IP-adres, het hardwareadres en de maximale verzendeenheid. Ook wordt er aangegeven of de interface actief is of niet.

In het volgende codevoorbeeld worden de NetworkInterface-eigenschappen van elke interface op de clientcomputer getraceerd:

```
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 );
                }
            }
        }
    }
}
}
```

Raadpleeg de volgende bronnen voor meer informatie:

- NetworkInfo
- NetworkInterface
- InterfaceAddress
- Flexpert: Detecting the network connection type with Flex 4.5

Wijzigingen in netwerkconnectiviteit

Adobe AIR 1.0 of hoger

Uw AIR-toepassing kan worden uitgevoerd in omgevingen met een onzekere of veranderende netwerkconnectiviteit. Om de verbindingen met online bronnen beter te kunnen beheren, verstuurt Adobe AIR de gebeurtenis network change wanneer een netwerkverbinding beschikbaar of onbeschikbaar wordt. De gebeurtenis `networkChange` wordt verzonden door zowel het NetworkInfo-object als het NativeApplication-object van de toepassing. Om op deze gebeurtenis te kunnen reageren, voegt u een listener toe:

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

Vervolgens definieert u een gebeurtenishandlerfunctie:

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

De `networkChange`-gebeurtenis geeft geen wijziging in alle netwerkactiviteit aan, maar alleen dat er een netwerkverbinding is gewijzigd. AIR probeert niet om de betekenis van de netwerkwijziging te interpreteren. Een computer in een netwerk kan vele reële en virtuele verbindingen hebben. Het verlies van een verbinding betekent dus niet noodzakelijk dat een resource verloren is gegaan. Anderzijds waarborgen nieuwe verbindingen ook geen betere beschikbaarheid van de resources. Soms kan een nieuwe verbinding zelfs de toegang blokkeren tot resources die voordien beschikbaar waren (bijvoorbeeld wanneer verbinding wordt gemaakt met een VPN).

Een toepassing kan alleen bepalen of deze verbinding kan maken met een externe resource door dit daadwerkelijk te proberen. Het framework voor servicebewaking biedt aan een opgegeven host een op gebeurtenissen gebaseerd middel om te reageren op wijzigingen in netwerkconnectiviteit.

Opmerking: *Het framework voor servicebewaking detecteert of een server op een aanvaardbare wijze reageert op een aanvraag. Een succesvolle controle garandeert geen volledige connectiviteit. Schaalbare webservices maken vaak gebruik van cachevoorzieningen en systemen om de belasting te verdelen, ten einde verkeer om te leiden naar een cluster van webservers. In deze situatie zorgen serviceproviders slechts voor een gedeeltelijke diagnose van de netwerkconnectiviteit.*

Servicebewaking

Adobe AIR 1.0 of hoger

Het framework voor servicebewaking staat los van het AIR-framework en bevindt zich in het bestand aircore.swc. Om het AIR-framework te kunnen gebruiken, moet het bestand aircore.swc zijn opgenomen in uw ontwikkelingsproces.

Adobe® Flash® Builder sluit deze bibliotheek automatisch in.

De klasse ServiceMonitor implementeert het framework voor de bewaking van netwerkservices en levert de basisfunctionaliteit voor de servicebewaking. Een instantie van de klasse ServiceMonitor verstuurt standaard gebeurtenissen met betrekking tot de netwerkconnectiviteit. Het ServiceMonitor-object verzendt deze gebeurtenissen als een instantie wordt gemaakt en als er door de runtime een netwerkwijziging wordt vastgesteld. Bovendien kunt u de eigenschap `pollInterval` van een ServiceMonitor-instantie instellen om de connectiviteit in een bepaald interval (in milliseconden) te controleren, ongeacht algemene gebeurtenissen betreffende de netwerkconnectiviteit. Een ServiceMonitor-object controleert de netwerkconnectiviteit pas als de methode `start()` is opgeroepen.

De klasse URLMonitor, een subklasse van de klasse ServiceMonitor, detecteert wijzigingen in de HTTP-connectiviteit voor een opgegeven URLRequest.

De klasse SocketMonitor, eveneens een subklasse van de klasse ServiceMonitor, detecteert wijzigingen in de connectiviteit met een opgegeven host op een opgegeven poort.

Opmerking: *Vóór AIR 2 is het servicecontroleframework gepubliceerd in de bibliotheek servicemonitor.swc. Deze bibliotheek is nu verouderd. Gebruik in plaats hiervan de bibliotheek aircore.swc.*

Flash CS4 en CS5 Professional

Ga als volgt te werk om deze klassen te gebruiken Adobe® Flash® CS4 of CS5 Professional:

1. Kies de opdracht Bestand > Instellingen publiceren.
2. Klik op de knop Instellingen voor ActionScript 3.0. Selecteer Bibliotheekpad.
3. Klik op de knop Bladeren naar SWC-bestand en blader naar de map AIK in de installatiemap van Flash Professional.
4. In deze map staan de bestanden /frameworks/libs/air/aircore.swc (voor AIR 2) of /frameworks/libs/air/servicemonitor.swc (voor AIR 1.5).
5. Klik op de knop OK.
6. Voeg de volgende import-instructie toe aan de ActionScript 3.0-code:

```
import air.net.*;
```

Flash CS3 Professional

Als u deze klassen in Adobe® Flash® CS3 Professional wilt gebruiken, moet u eerst de component ServiceMonitorShim vanuit het deelvenster Componenten naar de bibliotheek slepen. Voeg daarna de volgende `import`-instructie aan uw ActionScript 3.0-code toe:

```
import air.net.*;
```

HTTP-bewaking

Adobe AIR 1.0 of hoger

De klasse URLMonitor bepaalt of HTTP-aanvragen kunnen worden ingediend via een bepaald adres op poort 80 (de typische poort voor HTTP-communicatie). De volgende code gebruikt een instantie van de klasse URLMonitor om connectiviteitswijzigingen in de Adobe-website te detecteren:

```
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);
}
```

Socketbewaking

Adobe AIR 1.0 of hoger

AIR-toepassingen kunnen ook socketverbindingen gebruiken voor push-model connectiviteit. Om veiligheidsredenen beperken firewalls en netwerkrouters doorgaans de netwerkcommunicatie via ongeoorloofde poorten. Daarom moeten ontwikkelaars er rekening mee houden dat gebruikers niet altijd socketverbindingen tot stand kunnen brengen.

De volgende code gebruikt een instantie van de klasse SocketMonitor om connectiviteitswijzigingen in een socketverbinding te detecteren. De bewaakte poort is 6667, een veelgebruikte poort voor 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);
}
```

Als de socketserver een beveiligde verbinding vereist, kunt u de SecureSocketMonitor-klasse gebruiken in plaats van SocketMonitor.

DNS-records (Domain Name System)

Adobe AIR 2.0 of hoger

U kunt DNS-bronrecords opzoeken aan de hand van de klasse DNSResolver. DNS-bronrecords bieden informatie zoals het IP-adres van een domeinnaam en de domeinnaam van een IP-adres. U kunt de volgende DNS-bronrecordtypen opzoeken:

- ARecord - IPv4-adres voor een host.
- AAAARecord - IPv6-adres voor een host.
- MXRecord - mailuitwisselingsrecord voor een host.
- PTRRecord - hostnaam voor een IP-adres.
- SRVRecord - servicerecord voor een service..

Als u een record wilt opzoeken, geeft u een querytekenreeks en het klassenobject dat het recordtype representeert, door aan de methode `lookup()` van het DNSResolver-object. De te gebruiken querytekenreeks is afhankelijk van het recordtype:

Recordklasse	Querytekenreeks	Voorbeeldquerytekenreeks
ARecord	hostnaam	“example.com”
AAAARecord	hostnaam	“example.com”
MXRecord	hostnaam	“example.com”
PTRRecord	IP-adres	“208.77.188.166”
SRVRecord	Service-id: _service._protocol.host	“_sip._tcp.example.com”

In het volgende codevoorbeeld wordt het IP-adres van de host "example.com" opgezocht.

```
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 );
        }
    }
}
```

Raadpleeg de volgende bronnen voor meer informatie:

- DNSResolver

- DNSResolverEvent
- ARecord
- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

Hoofdstuk 43: Sockets

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een socket is een type netwerkverbinding tussen twee computerprocessen. Deze processen worden meestal uitgevoerd op twee verschillende computers die zijn gekoppeld aan hetzelfde IP-netwerk (Internet Protocol). De gekoppelde processen kunnen echter ook worden uitgevoerd op dezelfde computer via het speciale IP-adres “local host”.

Adobe Flash Player biedt ondersteuning voor TCP-sockets (Transport Control Protocol) van de client-kant. Een Flash Player-toepassing kan een verbinding maken met een ander proces dat handelt als een socketserver, maar de toepassing kan geen binnenkomende verbindingsaanvragen accepteren van andere processen. Met andere woorden: een Flash Player-toepassing kan verbinding maken met een TCP-server, maar niet functioneren als TCP-server.

De Flash Player-API bevat ook de XMLSocket-klasse. De XMLSocket-klasse gebruikt een Flash Player-specifiek protocol waarmee u XML-berichten kunt uitwisselen met een server die het protocol begrijpt. De XMLSocket-klasse werd in ActionScript 1 geïntroduceerd en wordt nog steeds ondersteund ten behoeve van compatibiliteit met oudere versies. In het algemeen dient u de klasse Socket te gebruiken voor nieuwe toepassingen, tenzij u een verbinding tot stand brengt met een server die specifiek is ingesteld op communicatie met Flash XMLSockets.

Adobe AIR voegt verschillende extra klassen toe voor netwerkprogrammering op basis van socket. Met de ServerSocket-klasse kunnen AIR-toepassingen fungeren als TCP-socketservers en verbinding maken met socketservers waarvoor SSL- of TLS-beveiliging is vereist. Verder kunnen AIR-toepassingen ook UDP-berichten (Universal Datagram Protocol) ontvangen en verzenden met de DatagramSocket-klasse.

Meer Help-onderwerpen

Het pakket flash.net

“Verbinding maken met sockets” op pagina 1136

TCP-sockets

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met TCP (het Transmission Control Protocol) kunt u berichten uitwisselen via een ononderbroken netwerkverbinding. TCP garandeert dat alle verzonden berichten in de juiste volgorde aankomen (behalve in geval van grote netwerkproblemen). Voor TCP-verbindingen is een "client" en een "server" vereist. Flash Player kan clientsockets maken. Bovendien kunt u in Adobe AIR serversockets maken.

De volgende ActionScript API's verschaffen TCP-verbindingen:

- Socket — hiermee kan een clienttoepassing verbinding maken met een server. De klasse Socket kan niet luisteren naar binnenkomende verbindingen.
- SecureSocket (AIR) — stelt een clienttoepassing in staat te verbinden met een betrouwbare server en gecodeerde communicatie uit te voeren.
- ServerSocket (AIR): stelt een toepassing in staat te luisteren naar binnenkomende verbindingen en te fungeren als een server.
- XMLSocket: hiermee kan een clienttoepassing verbinding maken met een XMLSocket-server.

Binaire clientsockets

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een binaire socketverbinding lijkt op een XML-socket behalve dat de client en server niet beperkt zijn tot het uitwisselen van XML-berichten. In plaats daarvan kan de verbinding gegevens als binaire informatie overbrengen. Hierdoor kunt u verbinding maken met meer soorten services, waaronder e-mailservers (POP3, SMTP en IMAP) en nieuwsservers (NNTP).

Socket, klasse

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse Socket kunt u socketverbindingen maken en onbewerkte binaire gegevens lezen en schrijven. De klasse Socket is nuttig wanneer u werkt met servers die binaire protocollen gebruiken. Als u binaire socketverbindingen gebruikt, kunt u code schrijven die interactie mogelijk maakt met verschillende internetprotocollen, zoals POP3, SMTP, IMAP en NNTP. Hierdoor kunnen uw toepassingen verbinding maken met e-mail- en nieuwsservers.

Flash Player kan verbinding maken met een server door direct gebruik te maken van het binaire protocol van die server. Sommige servers gebruiken de bytevolgorde big endian en sommige servers gebruiken de bytevolgorde little endian. De meeste servers op internet gebruiken de bytevolgorde big endian omdat de bytevolgorde van een netwerk big endian is. De bytevolgorde little-endian wordt veel gebruikt omdat de Intel® x86-architectuur deze gebruikt. Gebruik de endian-bytevolgorde die overeenkomt met de bytevolgorde van de server die gegevens verzendt of ontvangt. Alle bewerkingen die worden uitgevoerd door de interfaces IDataInput en IDataOutput, en de klassen die die interfaces implementeren (ByteArray, Socket en URLStream), worden standaard gecodeerd in de indeling big-endian, dat wil zeggen, met het meest significante byte eerst. Er is gekozen voor deze standaardbytevolgorde vanwege compatibiliteit met Java en de officiële netwerkbytevolgorde Als u de bytevolgorde wilt wijzigen en wilt opgeven dat big-endian of little-endian moet worden gebruikt, stelt u de eigenschap `endian` in op `Endian.BIG_ENDIAN` of `Endian.LITTLE_ENDIAN`.

De Socket-klasse neemt alle methoden over die worden gedefinieerd door de IDataInput- en IDataOutput-interfaces (te vinden in het pakket flash.utils). Deze methoden moeten worden gebruikt om te schrijven naar de socket en hieruit te lezen.

Raadpleeg de volgende bronnen voor meer informatie:

- Socket
- IDataInput
- IDataOutput
- socketData-gebeurtenis

Beveiligde clientsockets (AIR)

Adobe AIR 2 of hoger

U kunt de SecureSocket-klasse gebruiken om verbinding te maken met socketservers die gebruik maken van Secure Sockets Layer version 4 (SSLv4) of Transport Layer Security version 1 (TLSv1). Een beveiligde socket biedt drie voordelen: serververificatie, gegevensintegriteit en vertrouwelijkheid van berichten. De runtime verifieert een server aan de hand van het servercertificaat en de relatie van de server met de certificaten van de basis- of tussenliggende certificeringsinstantie in de vertrouwde opslag van de gebruiker. De runtime is afhankelijk van de cryptografiealgoritmen die door de implementaties van het SSL- en TLS-protocol worden gebruikt. Deze algoritmen bieden namelijk gegevensintegriteit en vertrouwelijkheid van berichten.

Als u met het SecureSocket-object verbinding maakt met een server, valideert de runtime het servercertificaat aan de hand van de vertrouwde certificaatopslag. Bij Windows en Mac wordt de vertrouwde opslag geleverd door het besturingssysteem. Bij Linux-computers wordt de vertrouwde opslag geleverd door de runtime.

Als het servercertificaat niet geldig is of niet wordt vertrouwd, verzendt de runtime een `ioError`-gebeurtenis. U kunt de eigenschap `serverCertificateStatus` van het SecureSocket-object controleren om te bepalen waarom de validatie is mislukt. Er is geen functionaliteit voor het communiceren met een server zonder een geldig en vertrouwd certificaat.

De klasse CertificateStatus definieert tekenreeksconstanten die de mogelijke validatieresultaten representeren:

- Vervallen - De vervaldatum van het certificaat is verstreken.
- Ongeldig - Er zijn een aantal redenen waarom een certificaat ongeldig kan zijn. Zo kan het certificaat gewijzigd of beschadigd zijn of kan het certificaat van het verkeerde type zijn.
- Ongeldige keten - Een of meer certificaten in de certificaatketen van de server is ongeldig.
- Niet-overeenkomende principal - De hostnaam van de server en de algemene naam van het certificaat komen niet overeen. Met andere woorden: de server gebruikt het verkeerde certificaat.
- Ingetrokken - De certificeringsinstantie die het certificaat heeft uitgegeven, heeft het certificaat ingetrokken.
- Vertrouwd - Het certificaat is geldig en wordt vertrouwd. Een SecureSocket-object kan alleen verbinding maken met een server die een geldig en vertrouwd certificaat gebruikt.
- Onbekend - Het SecureSocket-object heeft het certificaat nog niet gevalideerd. De eigenschap `serverCertificateStatus` heeft deze statuswaarde voordat u `connect()` aanroept en voordat een `connect`-gebeurtenis of `ioError`-gebeurtenis wordt verzonden.
- Niet-vertrouwde handtekeningen - Het certificaat wordt niet geketend aan een vertrouwd basiscertificaat in de vertrouwde opslag van de clientcomputer.

Bij communicatie met een SecureSocket-object moet de server een beveiligd protocol gebruiken en beschikken over een geldig en vertrouwd certificaat. In andere opzichten is het gebruik van een SecureSocket-object hetzelfde als het gebruik van een Socket-object.

Het SecureSocket-object wordt niet op alle platformen ondersteund. Gebruik de eigenschap `isSupported` van de klasse SecureSocket om te testen of het gebruik van het SecureSocket-object op de huidige clientcomputer door de runtime wordt ondersteund.

Raadpleeg de volgende bronnen voor meer informatie:

- SecureSocket
- CertificateStatus
- IDataInput
- IDataOutput
- socketData-gebeurtenis

Voorbeeld van TCP-socket: een Telnet-client maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In het Telnet-voorbeeld worden technieken geïllustreerd voor het maken van een verbinding met een externe server en het verzenden van gegevens met de klasse Socket. In dit voorbeeld worden de volgende technieken toegepast:

- Een aangepaste Telnet-client maken met behulp van de klasse Socket

- Tekst naar de externe server verzenden met een object ByteArray
- Ontvangen gegevens van een externe server verwerken

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. U vindt de bestanden voor de Telnet-toepassing in de map Samples/Telnet. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
TelnetSocket.fla of TelnetSocket.mxml	Het hoofdtoepassingsbestand dat bestaat uit de gebruikersinterface voor Flex (MXML) of Flash (FLA).
TelnetSocket.as	Klasse Document die de logica voor de gebruikersinterface bevat (alleen Flash).
com/example/programmingas3/Telnet/Telnet.as	Hiermee wordt de Telnet-clientfunctionaliteit voor de toepassing geboden, zoals verbinding maken met een externe server en gegevens verzenden, ontvangen en weergeven.

Overzicht van Telnet-sockettoepassing

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het hoofdbestand TelnetSocket.mxml maakt de gebruikersinterface (UI) voor de hele toepassing.

Naast de UI definieert dit bestand ook twee methoden, `login()` en `sendCommand()`, die de gebruiker verbinden met de opgegeven server.

De volgende code toont het ActionScript in het hoofdtoepassingsbestand:

```
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 = "";
}
```

De eerste coderegel importeert de klasse Telnet uit het aangepaste pakket com.example.programmingas.socket. De tweede coderegel declareert een instantie van de klasse Telnet, `telnetClient`, die later wordt geïnitialiseerd door de methode `connect()`. Vervolgens wordt de methode `connect()` gedeclareerd die de eerder gedeclareerde variabele `telnetClient` initialiseert. Deze methode geeft de door de gebruiker opgegeven naam voor de Telnet-server door, samen met de poort voor de Telnet-server en een verwijzing naar een TextArea-component in het weergaveoverzicht,

die wordt gebruikt om de tekstreacties van de socketserver weer te geven. Met de laatste twee regels van de methode `connect()` wordt de eigenschap `title` ingesteld voor Panel, waarna de component Panel wordt ingesteld. Hiermee kan de gebruiker gegevens naar de externe server verzenden. De laatste methode in het hoofdtoepassingsbestand, `sendCommand()`, wordt gebruikt om de opdrachten van de gebruiker als een object ByteArray naar de externe server te verzenden.

Overzicht van de klasse Telnet

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse Telnet is verantwoordelijk voor de verbinding met de externe Telnet-server en het verzenden/ontvangen van gegevens.

De klasse Telnet declareert de volgende variabelen van het type private:

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

De eerste variabele, `serverURL`, bevat het door de gebruiker opgegeven adres van de server waarmee verbinding moet worden gemaakt.

De tweede variabele, `portNumber`, is het poortnummer waarop de Telnet-server wordt uitgevoerd. De Telnet-service wordt standaard uitgevoerd op poort 23.

De derde variabele, `socket`, is een Socket-instantie die probeert verbinding te maken met de server die wordt aangegeven door de variabelen `serverURL` en `portNumber`.

De vierde variabele, `ta`, is een verwijzing naar een instantie van een TextArea-component in het werkgebied. Deze component wordt gebruikt om reacties van de externe Telnet-server of eventuele foutberichten weer te geven.

De laatste variabele, `state`, is een numerieke waarde die wordt gebruikt om te bepalen welke opties uw Telnet-client ondersteunt.

Zoals u eerder zag, wordt de constructorfunctie van de klasse Telnet aangeroepen door de methode `connect()` in het hoofdtoepassingsbestand.

De Telnet-constructor werkt met drie parameters: `server`, `port` en `output`. De parameters `server` en `port` geven de servernaam en het poortnummer op waarop de Telnet-server wordt uitgevoerd. De laatste parameter, `output`, is een verwijzing naar een instantie van een TextArea-component in het werkgebied waarin de uitvoer van de server voor de gebruikers wordt weergegeven.

```
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();
    }
}
```

Gegevens schrijven naar een socket

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u gegevens naar een socketverbinding wilt schrijven, kunt een van de schrijfmethoden in de klasse Socket aanroepen. Deze schrijfmethoden zijn onder meer `writeBoolean()`, `writeByte()`, `writeBytes()` en `writeDouble()`. Gebruik vervolgens de methode `flush()` om de gegevens in de uitvoerbuffer te verwijderen. In de Telnet-server worden gegevens naar de socketverbinding geschreven met de methode `writeBytes()`, die de bytearray als parameter neemt en naar de uitvoerbuffer verzendt. De methode `writeBytesToSocket()` ziet er als volgt uit:

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

Deze methode wordt aangeroepen door de methode `sendCommand()` van het hoofdtoepassingsbestand.

Berichten van de socketserver weergeven

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer een bericht wordt ontvangen van de socketserver of wanneer er een gebeurtenis plaatsvindt, wordt de aangepaste methode `msg()` aangeroepen. Deze methode voegt een tekenreeks toe aan de component TextArea in het werkgebied en roept de aangepaste methode `setScroll()` aan, die ervoor zorgt dat de component TextArea naar het einde schuift. De methode `msg()` ziet er als volgt uit:

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

Als u de inhoud van de component TextArea niet automatisch verschuift, zouden gebruikers handmatig de schuifbalken van het tekstgebied moeten verschuiven om het laatste antwoord van de server te zien.

Schuiven in een TextArea-component

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `setScroll()` bevat één regel ActionScript die de inhoud van de component TextArea verticaal verschuift, zodat de gebruiker de laatste regel van de geretourneerde tekst ziet. Het volgende fragment toont de methode `setScroll()`:

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

Deze methode stelt de eigenschap `verticalScrollPosition`, het regelnummer van de bovenste rij tekens die momenteel wordt weergegeven, in op de waarde van de eigenschap `maxVerticalScrollPosition`.

XML-sockets

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met een XML-socket kunt u een verbinding tot stand brengen met een externe server, waarbij deze verbinding geopend blijft totdat u deze expliciet sluit. U kunt reeksgegevens, zoals XML-gegevens, uitwisselen tussen de server en de client. Een voordeel van het gebruik van een XML-socketserver is dat de client gegevens niet expliciet hoeft aan te vragen. De server kan gegevens verzenden zonder op een aanvraag te moeten wachten en kan gegevens verzenden naar elke aangesloten client.

In Flash Player en in Adobe AIR-inhoud buiten de toepassingssandbox vereisen XML-socketverbindingen een socketbeleidbestand op de doelserver. Zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116 en "Verbinding maken met sockets" op pagina 1136 voor meer informatie.

De klasse XMLSocket kan niet automatisch een tunnel maken via firewalls omdat XMLSocket, in tegenstelling tot RTMP (Real-Time Messaging Protocol), niet beschikt over HTTP-tunnelmogelijkheden. Als u HTTP-tunnelmogelijkheden nodig hebt, kunt u overwegen Flash Remoting of Flash Media Server (met RTMP-ondersteuning) te gebruiken.

Voor inhoud in Flash Player of in een AIR-toepassing buiten de beveiligingssandbox van de toepassing gelden bepaalde beperkingen. Deze beperkingen zijn van toepassing op de manier waarop en de plaats waar de inhoud een XMLSocket-object kan gebruiken om verbinding te maken met de server:

- Voor inhoud buiten de toepassingsbeveiligingssandbox kan met de methode `XMLSocket.connect()` alleen verbinding worden gemaakt met TCP-poortnummers die hoger zijn dan of gelijk zijn aan 1024. Een gevolg van deze beperking is dat ook de serverdaemons die met het `XMLSocket`-object communiceren, moeten worden toegewezen aan poortnummers die hoger zijn dan of gelijk zijn aan 1024. Poortnummers die lager zijn dan 1024, worden vaak gebruikt door systeemservices, zoals FTP (21), Telnet (23), SMTP (25), HTTP (80) en POP3 (110), zodat XMLSocket-objecten uit veiligheidsoverwegingen van deze poorten worden buitengesloten. Het beperken van poortnummers verkleint de kans dat deze bronnen ongewenst worden geopend en misbruikt.

- Voor inhoud buiten de toepassingsbeveiligingssandbox kan met de methode `XMLSocket.connect()` alleen verbinding worden gemaakt met computers in hetzelfde domein als de inhoud. (Deze beperking is identiek aan de beveiligingsregels voor `URLLoader.load()`.) Als u verbinding wilt maken met een serverdaemon die wordt uitgevoerd in een ander domein dan het domein waarin de inhoud zich bevindt, kunt u op de server een domeinoverschrijdend beveiligingsbeleidsbestand maken dat toegang toestaat vanaf specifieke domeinen. Zie "Beveiliging in AIR" op pagina 1144 voor meer informatie over domeinoverschrijdende beleidsbestanden.

Opmerking: *Het kan lastig zijn een server zodanig in te stellen dat deze met het XMLSocket-object kan communiceren. Als uw toepassing geen realtime interactiviteit vereist, gebruikt u de klasse URLLoader in plaats van de klasse XMLSocket.*

U kunt de methoden `XMLSocket.connect()` en `XMLSocket.send()` van de klasse XMLSocket gebruiken om XML via een socketverbinding van en naar een server te verzenden. Met de methode `XMLSocket.connect()` wordt een socketverbinding met een webserverpoort tot stand gebracht. Met de methode `XMLSocket.send()` wordt een XML-object doorgegeven aan de server die is opgegeven in de socketverbinding.

Als u de methode `XMLSocket.connect()` oproept, opent de toepassing een TCP/IP-verbinding met de server. Deze verbinding wordt opengehouden totdat een van de volgende gebeurtenissen plaatsvindt:

- De methode `XMLSocket.close()` van de klasse XMLSocket wordt opgeroepen.
- Er bestaan geen verwijzingen meer naar het XMLSocket-object.
- Flash Player wordt afgesloten.
- De verbinding wordt verbroken (de modemverbinding wordt bijvoorbeeld verbroken).

Verbinding maken met een server met behulp van de XMLSocket-klasse

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een socketverbinding wilt maken, moet u een servertoepassing maken die wacht op de aanvraag voor de socketverbinding en een antwoord verzendt naar de Flash Player- of AIR-toepassing. Dit soort servertoepassingen kan worden geschreven in AIR of in een andere programmeertaal zoals Java, Python of Perl. Als u de klasse XMLSocket wilt gebruiken, moet de servercomputer een daemon uitvoeren waarmee het eenvoudige protocol kan worden herkend dat door de klasse XMLSocket wordt gebruikt:

- XML-berichten worden via een full-duplex TCP/IP-streamsocketverbinding verzonden.
- Elk XML-bericht is een volledig XML-document dat met een nulbyte (0) wordt afgesloten.
- Er kan een onbeperkt aantal XML-berichten worden verzonden en ontvangen via één XMLSocket-verbinding.

Java-XML-socketservers maken en er verbinding mee maken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De volgende code bevat een voorbeeld van een eenvoudige XMLSocket-server, geschreven in Java, die inkomende verbindingen accepteert en de ontvangen berichten weergeeft in het opdrachtpromptvenster. Standaard wordt een nieuwe server gemaakt op poort 8080 van de lokale computer, maar u kunt desgewenst een ander poortnummer opgeven wanneer u de server vanaf de opdrachtregel start.

Maak een nieuwe tekstdocument en voeg de volgende code toe:

```
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);
    }
}
```

Sla het document op de vaste schijf op als SimpleServer.java en compileer dit met een Java-compiler. Hierbij wordt een Java-klassebestand met de naam SimpleServer.class gemaakt.

U kunt de XMLSocket-server starten door een opdrachtprompt te openen en `java SimpleServer` te typen. Het bestand SimpleServer.class kan overal op de lokale computer of in het netwerk worden opgeslagen; het hoeft niet in de hoofdmap van de webserver te worden geplaatst.

Als u de server niet kunt starten omdat de bestanden zich niet in het Java-klassepad bevinden, kunt u de server proberen te starten met `java -classpath. SimpleServer`.

Als u verbinding wilt maken met de XMLSocket vanuit uw toepassing, moet u een nieuwe instantie van de XMLSocket-klasse maken en de methode `XMLSocket.connect()` aanroepen, waarbij u een hostnaam en poortnummer doorgeeft, zoals in de volgende code:

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

Wanneer u gegevens van de server ontvangt, wordt de gebeurtenis `data` (`flash.events.DataEvent.DATA`) verzonden:

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

Als u gegevens wilt verzenden naar de XMLSocket-server, gebruikt u de methode `XMLSocket.send()` en geeft u een XML-object of tekenreeks door. Flash Player zet de opgegeven parameter om in een object String en verzendt de inhoud naar de XMLSocket-server, gevolgd door een nulbyte (0):

```
xmlsock.send(xmlFormattedData);
```

De methode `XMLSocket.send()` retourneert geen waarde die aangeeft of de gegevens zijn verzonden. Als er een fout optreedt terwijl wordt geprobeerd gegevens te verzenden, wordt een IOError-fout gegenereerd.

Elk bericht dat u naar de XML-socketserver verzendt, moet eindigen op een nieuwe-regelteken (`\n`).

Zie XMLSocket voor meer informatie.

Serversockets

Adobe AIR 2 of hoger

Gebruik de klasse ServerSocket om toe te staan dat andere processen met gebruik van een TCP-socket (Transport Control Protocol) verbinding maken met uw toepassing. Het proces dat een verbinding maakt, kan worden uitgevoerd op een lokale computer of op een computer met een netwerkverbinding. Als een ServerSocket-object een verbindingsaanvraag ontvangt, verzendt dit object een `connect`-gebeurtenis. Het ServerSocketConnectEvent-object dat bij deze gebeurtenis wordt verzonden, bevat een Socket-object. U kunt dit Socket-object gebruiken voor verdere communicatie met het andere proces.

Luisteren of er socketverbindingen binnenkomen:

1. Maak een ServerSocket-object en zorg dat dit gebonden is aan een lokale poort.
2. Voeg gebeurtenislisteners voor de gebeurtenis `connect` toe.
3. Roep de methode `listen()` aan.
4. Reageer op de gebeurtenis `connect`. Deze gebeurtenis levert voor elke binnenkomende gebeurtenis een Socket-object.

Het ServerSocket-object blijft luisteren of er nieuwe verbindingen worden gemaakt totdat u de methode `close()` aanroept.

In het volgende codevoorbeeld wordt getoond hoe u een socketservertoepassing kunt maken. In dit voorbeeld wordt er geluisterd of er verbindingen binnenkomen op poort 8087. Als een verbinding wordt ontvangen, wordt er een bericht verzonden (de tekenreeks "Connected") naar de clientsocket. Hierna verzendt de server een herhaling van de ontvangen berichten terug naar de 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." );
        }
}}
```

Raadpleeg de volgende bronnen voor meer informatie:

- ServerSocket
- ServerSocketConnectEvent
- Socket

UDP-sockets (AIR)

Adobe AIR 2 of hoger

Met het Universal Datagram Protocol (UDP) kunt u berichten uitwisselen via een staatloze netwerkverbinding. UDP biedt geen garantie dat berichten op volgorde worden afgeleverd of dat berichten überhaupt worden afgeleverd. Bij het gebruik van UDP hoeft de netwerkcode van het besturingssysteem doorgaans minder aandacht te besteden aan het marshallen, bijhouden en bevestigen van berichten. UDP-berichten arriveren daarom meestal met minder vertraging op hun bestemming dan TCP-berichten.

UDP-socketcommunicatie is handig als u realtime-informatie moet verzenden, bijvoorbeeld positie-updates tijdens een game of geluidspakketen bij een audiochattoepassing. Bij dergelijke toepassingen is een bepaalde mate van gegevensverlies acceptabel en is een kleine vertraging bij het verzenden belangrijker dan een gegarandeerde aflevering. Voor bijna alle andere doeleinden zijn TCP-sockets een betere keuze.

Uw AIR-toepassing kan UDP-berichten verzenden en ontvangen met de klassen DatagramSocket en DatagramSocketDataEvent. Een UDP-bericht verzenden of ontvangen:

1. Maak een DatagramSocket-object.
2. Voeg een gebeurtenislistener voor de gebeurtenis `data` toe.
3. Gebruik de methode `bind()` om de socket aan een lokaal IP-adres en een lokale IP-poort te binden.
4. Verzend berichten door de methode `send()` aan te roepen en geef hierbij het IP-adres en de IP-poort van de lokale computer door.

5 Ontvang berichten door te reageren op de gebeurtenis `data`. Het DatagramSocketDataEvent-object dat voor deze gebeurtenis wordt verzonden, bevat een ByteArray-object met de berichtgegevens.

In het volgende codevoorbeeld wordt getoond hoe een toepassing UDP-berichten kan verzenden en ontvangen. In het voorbeeld wordt één bericht met de tekenreeks "Hello" naar de doelcomputer verstuurd. Ook wordt de inhoud van ontvangen berichten getraceerd.

```
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 ) );
    }
}}
```

Houd bij het gebruik van UDP-sockets rekening met de volgende overwegingen:

- Eén gegevenspakket kan niet groter zijn dan de kleinste maximale verzendeenheid van de netwerkinterface of van de netwerknodes tussen de zender en de ontvanger. Alle gegevens in het ByteArray-object dat aan de methode send() wordt doorgegeven, worden als één enkel pakket verzonden. (Bij TCP worden grote berichten verdeeld in afzonderlijke pakketten.)
- Er vindt geen handshake plaats tussen de zender en de ontvanger. Berichten worden zonder waarschuwing verwijderd als de ontvanger niet bestaat of als er zich bij de ontvanger geen actieve listener op de opgegeven poort bevindt.
- Als u de methode `connect()` gebruikt, worden door andere bronnen verzonden berichten genegeerd. Een UDP-verbinding biedt alleen passende pakketfiltering. Dit hoeft niet te betekenen dat er zich een geldige listener op het adres en de poort van de ontvanger bevindt.
- Een netwerk kan overbelast raken door UDP-verkeer. Als dit gebeurt, is het wellicht nodig dat netwerkbeheerders maatregelen implementeren om de servicekwaliteit te bewaken. (TCP beschikt over een ingebouwd controlemechanisme voor netwerkverkeer, zodat de gevolgen van een overbelast netwerk beperkt blijven.)

Raadpleeg de volgende bronnen voor meer informatie:

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

IPv6-adressen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player 9.0.115.0 en hoger ondersteunen IPv6 (Internet Protocol versie 6). IPv6 is een versie van Internet Protocol die 128-bits adressen ondersteunt (een verbetering ten opzichte van het eerdere IPv4-protocol dat alleen 32-bits adressen ondersteunt). Mogelijk moet u IPv6 activeren op uw netwerkinterfaces. Raadpleeg de Help van het besturingssysteem dat als host van de gegevens fungeert voor meer informatie.

Als IPv6 wordt ondersteund door het hostsysteem, kunt u numerieke literale IPv6-adressen opgeven in URL's tussen haakjes ([]), bijvoorbeeld:

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

Flash Player retourneert literale IPv6-waarden die aan de volgende regels voldoen:

- Flash Player retourneert de lange vorm van de tekenreeks voor IPv6-adressen.
- De IP-waarde heeft geen afkortingen met twee dubbele punten.
- Hexadecimale cijfers zijn alleen in kleine letters.
- IPv6-adressen staan tussen vierkante haken ([]).
- Elk adresviertal wordt weergegeven als 0 tot 4 hexadecimale cijfers; voorloopnullen worden weggelaten.
- Een adresviertal van alleen nullen wordt weergegeven als één enkele nul (geen twee dubbele punten), behalve als deze voorkomt in de volgende lijst met uitzonderingen.

De IPv6-waarden die Flash Player retourneert, hebben de volgende uitzonderingen:

- Een niet-opgegeven IPv6-adres (alleen nullen) wordt weergegeven als [::].

- Het IPv6-adres van de loopback of localhost wordt weergegeven als [::1].
- Aan IPv4 toegewezen adressen (geconverteerd naar IPv6) worden weergegeven als [::ffff:a.b.c.d], waarbij a.b.c.d een gewone decimale IPv4-waarde met punten is.
- IPv4-compatibele adressen worden weergegeven als [::a.b.c.d], waarbij a.b.c.d een gewone decimale IPv4-waarde met punten is.

Hoofdstuk 44: HTTP-communicatie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassingen Adobe® AIR® en Adobe® Flash® Player communiceren met HTTP-servers om gegevens, afbeeldingen, en video te laden en om berichten uit te wisselen.

Meer Help-onderwerpen

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

Externe gegevens laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

ActionScript 3.0 biedt manieren om gegevens uit externe bronnen te laden. Die bronnen kunnen statische inhoud bieden, zoals tekstbestanden, of dynamische inhoud die is gegenereerd door een webscript. De gegevens kunnen verschillende indelingen hebben en ActionScript biedt functionaliteit voor het decoderen en het openen van de gegevens. Tijdens het ophalen van gegevens kunt u ook gegevens naar de externe server verzenden.

De klasse URLRequest gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Veel API's die externe gegevens laden gebruiken de URLRequest-klasse om de eigenschappen van benodigde netwerkaanvragen te definiëren.

URLRequest-eigenschappen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de volgende eigenschappen van een URLRequest-object instellen in elke beveiligingssandbox:

Eigenschap	Beschrijving
contentType	Dit is het MIME-inhoudstype van alle gegevens die met de URL-aanvraag worden verzonden. Als er geen waarde voor contentType is ingesteld, worden waarden verzonden als `application/x-www-form-urlencoded`.
data	Een object dat gegevens bevat die met de URL-aanvraag moeten worden verzonden.
digest	Een tekenreeks waarmee het getekende Adobe-platformonderdeel wordt geïdentificeerd dat moet worden opgeslagen in (of opgehaald uit) de Adobe® Flash® Player-cache.
method	De HTTP-aanvraagmethode, zoals GET of POST. (Inhoud die in het beveiligingsdomein van de AIR-toepassing wordt uitgevoerd, kan andere tekenreeksen opgeven dan `"GET"` of `"POST"` voor de eigenschap `method`. Elke HTTP-term is toegestaan en `"GET"` is de standaardmethode. Zie "Beveiliging in AIR" op pagina 1144.)
requestHeaders	De array van HTTP-aanvraagheaders die aan de HTTP-aanvraag moet worden toegevoegd. Merk op dat de machtigingen voor het instellen van bepaalde headers beperkt zijn in Flash Player en in AIR-inhoud die wordt uitgevoerd buiten de beveiligingssandbox van de toepassing.
url	Hiermee geeft u de URL op die moet worden aangevraagd.

In AIR kunt u aanvullende eigenschappen van de URLRequest-klasse instellen die alleen beschikbaar zijn voor AIR-inhoud die wordt uitgevoerd in de beveiligingssandbox van de toepassing. Inhoud in de toepassingssandbox kan ook URL's definiëren met behulp van nieuwe URL-schema's (die een aanvulling vormen op de standaardschema's zoals `file` en `http`).

Eigenschap	Beschrijving
followRedirects	Hiermee geeft u aan of omleidingen moeten worden gevolgd (`true`, de standaardwaarde) of niet (`false`). Deze eigenschap wordt alleen ondersteund in de sandbox van de AIR-toepassing.
manageCookies	Hiermee geeft u voor deze aanvraag aan of de HTTP-protocolstack cookies moet beheren (`true`, de standaardwaarde) of niet (`false`). Het instellen van deze eigenschap wordt alleen ondersteund in de sandbox van de AIR-toepassing.
authenticate	Hiermee geeft u aan of verificatieverzoeken voor deze aanvraag moeten worden afgehandeld (`true`). Het instellen van deze eigenschap wordt alleen ondersteund in de sandbox van de AIR-toepassing. Standaard worden aanvragen geverifieerd. Dit kan ertoe leiden dat een verificatiedialoogvenster wordt weergegeven als de server referenties vereist. U kunt de gebruikersnaam en het wachtwoord ook instellen met de URLRequestDefaults-klasse. Zie "Standaardwaarden voor URLRequest instellen (alleen voor AIR)" op pagina 860.
cacheResponse	Hiermee geeft u aan of antwoordgegevens voor deze aanvraag in de cache moeten worden geplaatst. Het instellen van deze eigenschap wordt alleen ondersteund in de sandbox van de AIR-toepassing. Het antwoord wordt standaard in de cache geplaatst (`true`).
useCache	Hiermee geeft u aan of de lokale cache moet worden geraadpleegd voordat gegevens worden opgehaald door deze URLRequest. Het instellen van deze eigenschap wordt alleen ondersteund in de sandbox van de AIR-toepassing. Als de standaardwaarde (`true`) wordt aangehouden, wordt de lokale versie in de cache gebruikt, indien deze beschikbaar is.
userAgent	Hiermee geeft u de tekenreeks van de gebruikersagent op die in de HTTP-aanvraag moet worden gebruikt.

Opmerking: *De klasse HTMLLoader heeft verwante eigenschappen voor instellingen die betrekking hebben op inhoud die wordt geladen door een HTMLLoader-object. Zie "Informatie over de klasse HTMLLoader" op pagina 1039 voor meer informatie.*

Standaardwaarden voor URLRequest instellen (alleen voor AIR)

Adobe AIR 1.0 of hoger

Met de klasse URLRequestDefaults kunt u de toepassingsspecifieke standaardwaarden van URLRequest-objecten definiëren. Met de volgende code worden bijvoorbeeld de standaardwaarden van de eigenschappen `manageCookies` en `useCache` ingesteld. De opgegeven waarden voor deze eigenschappen worden door alle nieuwe URLRequest-objecten gebruikt, in plaats van de normale standaardwaarden:

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

Opmerking: *De URLRequestDefaults-klasse wordt alleen gedefinieerd voor inhoud die in Adobe AIR wordt uitgevoerd. Deze klasse wordt niet ondersteund in Flash Player.*

De klasse URLRequestDefaults bevat de methode `setLoginCredentialsForHost()`, waarmee u een standaard gebruikersnaam en een standaardwachtwoord kunt opgeven die moeten worden gebruikt voor een specifieke host. De host, die wordt gedefinieerd met de parameter hostname van de methode, kan bestaan uit een domein, zoals `"www.example.com"`, of uit een domein en een poortnummer, zoals `"www.example.com:80"`. Houd er rekening mee dat `"example.com"`, `"www.example.com"` en `"sales.example.com"` elk als een unieke host worden beschouwd.

Deze gebruikersgegevens worden alleen gebruikt als de server deze vereist. Als de gebruiker al is geverifieerd (bijvoorbeeld door het dialoogvenster voor verificatie te gebruiken), wordt de geverifieerde gebruiker niet gewijzigd door de methode `setLoginCredentialsForHost()` aan te roepen.

Met de volgende code wordt de standaardgebruikersnaam en -wachtwoord ingesteld voor aanvragen die naar www.example.com worden verzonden:

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

De URLRequestDefaults-instellingen zijn alleen van toepassing op het huidige toepassingsdomein, met één uitzondering. De verificatiegegevens die zijn doorgegeven aan de methode `setLoginCredentialsForHost()` worden gebruikt voor aanvragen die in elk willekeurig toepassingsdomein zijn gemaakt met de AIR-toepassing.

Zie de klasse URLRequestDefaults in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform *voor meer informatie.*

URI-schema's

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De standaard-URI-schema's, zoals de onderstaande, kunnen worden gebruikt bij aanvragen die worden gemaakt vanuit een willekeurige beveiligingssandbox:

http: en https:
U kunt deze schema's gebruiken voor standaardinternet-URL's (op dezelfde manier waarop ze worden gebruikt in een webbrowser).

file:
Gebruik `file:` om de URL aan te geven van een bestand dat zich in het lokale bestandssysteem bevindt. Bijvoorbeeld:

```
file:///c:/AIR Test/test.txt
```

U kunt in AIR ook de volgende schema's gebruiken wanneer u een URL definieert voor inhoud die in de beveiligingssandbox van de toepassing wordt uitgevoerd:

app:
Gebruik `app:` om een pad op te geven dat betrekking heeft op de hoofdmap van de geïnstalleerde toepassing. Het volgende pad verwijst bijvoorbeeld naar de submap resources van de map met de geïnstalleerde toepassing:

```
app:/resources
```

Wanneer een AIR-toepassing wordt gestart met de ADL (AIR Debug Launcher), is de toepassingsmap de map waarin het descriptorbestand van de toepassing zich bevindt.

Voor de URL (en de eigenschap `url`) voor een File-object dat met `File.applicationDirectory` is gemaakt, wordt het URI-schema `app` gebruikt, zoals hierna wordt geïllustreerd:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:
Gebruik `app-storage:` om een pad op te geven dat betrekking heeft op de gegevensopslagmap van de toepassing. Voor elke geïnstalleerde toepassing (en voor elke gebruiker) maakt AIR een unieke toepassingsopslagmap. Dit is een nuttige plaats om gegevens op te slaan die specifiek voor deze toepassing. Het volgende pad verwijst bijvoorbeeld naar het bestand prefs.xml in de submap Settings van de opslagmap van een toepassing:

```
app-storage:/settings/prefs.xml
```

Voor de URL (en de eigenschap `url`) voor een File-object dat met `File.applicationStorageDirectory` is gemaakt, wordt het URI-schema `app-storage` gebruikt, zoals hierna wordt geïllustreerd:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto:
U kunt het mailto-schema gebruiken in URLRequest-objecten die worden doorgegeven naar de functie `navigateToURL()`. Zie “Een URL openen in een andere toepassing” op pagina 875.

U kunt een URLRequest-object dat een van deze URI-schema's gebruikt, gebruiken om de URL-aanvraag te definiëren voor een aantal verschillende objecten, zoals een FileStream- of Sound-object. U kunt deze schema's ook gebruiken in HTML-inhoud die in AIR wordt uitgevoerd. Zo kunt u de schema's gebruiken in het kenmerk `src` van de tag `img`.

U kunt deze AIR-specifieke URI-schema's (`app:` en `app-storage:`) echter alleen gebruiken in inhoud in de beveiligingssandbox van de toepassing. Zie “Beveiliging in AIR” op pagina 1144 voor meer informatie.

URL-variabelen instellen

Het is natuurlijk mogelijk om variabelen direct toe te voegen aan de URL-tekenreeks, maar vaak is het gemakkelijker om de variabelen die nodig zijn voor een aanvraag te definiëren met de URLVariables-klasse.

U kunt op drie manieren parameters toevoegen aan een URLVariables-object:

- Binnen de URLVariables-constructor
- Met de methode `URLVariables.decode()`
- Als dynamische eigenschappen van het URLVariables-object zelf

In het volgende voorbeeld worden de drie methoden geïllustreerd. Ook ziet u hoe u de variabelen kunt toewijzen aan een URLRequest-object:

```
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;
```

Wanneer u variabelen definieert in de URLVariables-constructor of in de `URLVariables.decode()`-methode, moet u de tekens die een speciale betekenis hebben in een URI-tekenreeks coderen met een URL-code. Als u bijvoorbeeld een en-teken (&) gebruikt in een parameternaam of -waarde, moet u dit teken coderen door het te wijzigen van `&` in`%26`. Het en-teken fungeert namelijk als scheidingsteken voor parameters. Hiervoor kunt u de `encodeURIComponent()`-functie op het bovenste niveau gebruiken.

De klasse URLLoader gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de URLLoader -klasse kunt u een aanvraag indienen bij een server en hebt u toegang tot de geretourneerde informatie. Met de URLLoader-klasse hebt u ook toegang tot bestanden in het lokale bestandssysteem in contexten waar toegang tot lokale bestanden is toegestaan (zoals bij de Flash Player-sandbox Lokaal-met-bestandssysteem en de AIR-toepassingssandbox). De klasse URLLoader downloadt gegevens via een URL als tekst, binaire gegevens of URL-gecodeerde variabelen. De klasse URLLoader verzendt gebeurtenissen zoals `complete`, `httpStatus`, `ioError`, `open`, `progress` en `securityError`.

Het gebeurtenisafhandelingsmodel van ActionScript 3.0 is aanzienlijk anders dan het model van ActionScript 2.0, dat gebruikmaakte van de gebeurtenishandlers `LoadVars.onData`, `LoadVars.onHTTPStatus` en `LoadVars.onLoad`. Zie "Gebeurtenissen afhandelen" op pagina 133 voor meer informatie over gebeurtenisafhandeling in ActionScript 3.0.

De gedownloade gegevens zijn pas beschikbaar nadat het downloaden is voltooid. U kunt de downloadvoortgang (geladen bytes en totale aantal bytes) volgen door te luisteren naar de gebeurtenis `progress` die moet worden verzonden. Als een bestand snel wordt geladen, is het mogelijk dat de gebeurtenis `progress` niet wordt verzonden. Wanneer een bestand zonder problemen is gedownload, wordt de gebeurtenis `complete` verzonden. Als u de eigenschap URLLoader `dataFormat` instelt, ontvangt u de gegevens als tekst, als binaire gegevens met de indeling RAW of als een URLVariables-object.

De methode `URLLoader.load()` (en eventueel de constructor van de klasse URLLoader) gebruikt één parameter, namelijk `request`. Deze parameter is een URLRequest-object. Een URLRequest-object bevat alle informatie voor één HTTP-aanvraag, zoals de doel-URL, de aanvraagmethode (`GET` of `POST`), aanvullende headerinformatie en het MIME-type.

Als u bijvoorbeeld een XML-pakket naar een script op de server wilt uploaden, kunt u de volgende code gebruiken:

```
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);
```

Het voorgaande codefragment maakt een XML-document met de naam `dataXML` die het XML-pakket bevat dat naar de server moet worden gezonden. In het voorbeeld wordt de eigenschap URLRequest `contentType` ingesteld op `"text/xml"` en wordt het XML-document toegewezen aan de URLRequest `data`-eigenschap. Tot slot wordt in het voorbeeld een URLLoader-object gemaakt, waarna de aanvraag naar het externe script wordt verzonden met de methode `load()`.

De klasse URLStream gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De URLStream-klasse biedt tijdens het downloaden toegang tot de gegevens. Met de klasse URLStream kunt u bovendien een stream sluiten voordat deze geheel is gedownload. De gedownloade gegevens zijn beschikbaar als binaire gegevens met de indeling RAW.

Tijdens het lezen van de gegevens van een URLStream-object moet u de eigenschap `bytesAvailable` gebruiken om na te gaan of er voldoende gegevens zijn om te worden gelezen. Als u meer gegevens wilt lezen dan er beschikbaar zijn, treedt een EOFError-uitzondering op.

De httpResponseStatus-gebeurtenis (AIR)
In Adobe AIR verzendt de URLStream-klasse ook een `httpResponseStatus`-gebeurtenis (naast de `httpStatus`-gebeurtenis). De `httpResponseStatus`-gebeurtenis wordt vóór eventuele responsgegevens verzonden. De gebeurtenis `httpResponseStatus` (vertegenwoordigd door de HTTPStatusEvent-klasse) bevat de eigenschap `responseURL`. Dit is de URL van waaruit het antwoord is geretourneerd. De gebeurtenis bevat ook de eigenschap `responseHeaders`. Dit is een array van URLRequestHeader-objecten die de antwoordheaders vertegenwoordigen die door het antwoord zijn geretourneerd.

Gegevens uit externe documenten laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u dynamische toepassingen maakt, kan het nuttig zijn om gegevens te laden uit externe bestanden of serverscripts. Hiermee kunt u dynamische toepassingen maken zonder dat u uw toepassingen moet bewerken of opnieuw compileren. Als u bijvoorbeeld een toepassing voor een 'tip van de dag' maakt, kunt u een serverscript schrijven waarmee een willekeurige tip uit een database wordt opgehaald en eenmaal per dag wordt opgeslagen in een tekstbestand. Vervolgens kan uw toepassing de inhoud van een statisch tekstbestand laden in plaats van elke keer opnieuw query's uit te voeren.

Het volgende codefragment maakt een URLRequest- en URLLoader-object dat de inhoud van een extern tekstbestand, params.txt, laadt:

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

Als u geen aanvraagmethode definieert, laden Flash Player en Adobe AIR de inhoud standaard met de HTTP-methode `GET`. Als u de aanvraag wilt verzenden met de methode `POST`, stelt u de eigenschap `request.method` in op `POST` met behulp van de statische constante `URLRequestMethod.POST`, zoals wordt weergegeven in de volgende code:

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

Het externe document params.txt dat tijdens de runtime wordt geladen, bevat de volgende gegevens:

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,Dece
mber&dayNames=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Het bestand bevat twee parameters: `monthNames` en `dayNames`. Elke parameter bevat een door komma's gescheiden lijst die als tekenreeksen wordt geparseerd. U kunt deze lijst in een array splitsen met de methode `String.split()`.

Gebruik geen gereserveerde woorden of taalconstructies als variabelenamen in externe gegevensbestanden, omdat de code hierdoor moeilijker kan worden gelezen en fouten moeilijker kunnen worden opgespoord.

Zodra de gegevens zijn geladen, wordt de gebeurtenis `complete` verzonden en is de inhoud van het externe document beschikbaar voor gebruik in de eigenschap `data` van de URLLoader, zoals wordt weergegeven in de volgende code:

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

Als het externe document naam-waardeparen bevat, kunt u de gegevens parseren met de klasse URLVariables door de inhoud van het geladen bestand als volgt door te geven:

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

Elk naam-waardepaar uit het externe bestand wordt gemaakt als een eigenschap in het URLVariables-object. Elke eigenschap in het variabelenobject in het voorgaande codevoorbeeld wordt als een tekenreeks behandeld. Als de waarde van het naam-waardepaar een lijst met items is, kunt u de tekenreeks in een array omzetten door als volgt de methode `String.split()` op te roepen:

```
var dayNameArray:Array = variables.dayNames.split(",");
```

Als u numerieke gegevens laadt uit externe tekstbestanden, moet u de waarden omzetten in numerieke waarden door gebruik te maken van een functie op hoofdniveau, zoals `int()`, `uint()` of `Number()`.

In plaats van de inhoud van het externe bestand als tekenreeks te laden en een nieuwe object URLVariables te maken, zou u de eigenschap `URLLoader.dataFormat` kunnen instellen op een van de statische eigenschappen in de klasse URLLoaderDataFormat. De drie mogelijke waarden voor de eigenschap `URLLoader.dataFormat` zijn:

- `URLLoaderDataFormat.BINARY`: de eigenschap `URLLoader.data` bevat binaire gegevens die zijn opgeslagen in een object ByteArray.
- `URLLoaderDataFormat.TEXT`: de eigenschap `URLLoader.data` bevat tekst in een object String.
- `URLLoaderDataFormat.VARIABLES`: de eigenschap `URLLoader.data` bevat URL-gecodeerde variabelen die zijn opgeslagen in een object URLVariables.

In de volgende code ziet u hoe u gegevens die in een object URLVariables zijn opgeslagen, automatisch kunt parseren door de eigenschap `URLLoader.dataFormat` in te stellen op `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);
        }
    }
}
```

Opmerking: *De standaardwaarde voor* `URLLoader.dataFormat` *is* `URLLoaderDataFormat.TEXT`.

Het laden van XML uit een extern bestand staat gelijk aan het laden van URLVariables, zoals in het volgende voorbeeld wordt geïllustreerd. U kunt een URLRequest- en een URLLoader-instantie maken en deze gebruiken om een extern XML-document te downloaden. Wanneer het bestand volledig is gedownload, wordt de gebeurtenis `Event.COMPLETE` verzonden, waarna de inhoud van het externe bestand wordt omgezet in een XML-instantie, die u kunt parseren met XML-methoden en -eigenschappen.

```
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());
        }
    }
}
```

Communiceren met externe scripts

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse URLVariables niet alleen gebruiken om externe gegevensbestanden te laden, maar ook om variabelen te verzenden naar een serverscript en de reactie van de server te verwerken. Dit is bijvoorbeeld handig als u een spel programmeert en de score van de gebruiker naar een server wilt verzenden, waarna kan worden berekend of de gebruiker aan de lijst met topscores moet worden toegevoegd. U kunt zelfs de aanmeldingsgegevens van de gebruiker voor verificatie naar een server verzenden. Een serverscript kan de gebruikersnaam en het wachtwoord verwerken, deze met een database vergelijken en een bevestiging terugsturen waarin wordt aangegeven of de gebruikersgegevens geldig zijn.

In het volgende codefragment wordt een object URLVariables met de naam `variables` gemaakt, dat een nieuwe variabele met de naam `name` maakt. Vervolgens wordt een URLRequest-object gemaakt dat de URL aangeeft van het serverscript waar de variabelen naartoe moeten worden verzonden. Vervolgens stelt u de eigenschap `method` van het URLRequest-object zo in dat de variabelen als de HTTP-aanvraag `POST` worden verzonden. U voegt het URLVariables-object aan de URL-aanvraag toe door de eigenschap `data` van het URLRequest-object in te stellen op het URLVariables-object dat u eerder hebt gemaakt. Tot slot wordt de URLLoader-instantie gemaakt en wordt de methode `URLLoader.load()` opgeroepen, die de aanvraag activeert.

```
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);
}
```

De volgende code bevat de inhoud van het Adobe ColdFusion®-document greeting.cfm, dat in het vorige voorbeeld wordt gebruikt:

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

Webserviceaanvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er bestaan verschillende HTPP-webservices. De hoofdtypen zijn:

- REST
- XML-RPC
- SOAP

Als u een webservice in ActionScript 3 wilt gebruiken, maakt u eerst een URLRequest-object. Vervolgens construeert u de webserviceaanroep met URL-variabelen of een XML-document en tot slot verzendt u de aanroep naar de service met behulp van een URLLoader-object. Het Flex-raamwerk bevat verschillende klassen om het gebruik van webservices gemakkelijker te maken. Dit is vooral handig bij toegang tot complexe SOAP-services. Vanaf Flash Professional CS3 kunt u de Flex-klassen gebruiken in toepassingen die zijn ontwikkeld met Flash Professional en met Flash Builder.

Bij AIR-toepassingen die zijn gebaseerd op HTML kunt u enerzijds de URLRequest- en URLLoader-klassen gebruiken, of anderzijds de XMLHttpRequest-klasse van Javascript. Indien gewenst, kunt u ook een SWF-bibliotheek maken waarmee de webservicecomponenten van het Flex-framework beschikbaar worden gemaakt voor uw Javascript-code.

Wanneer uw toepassing in een browser wordt uitgevoerd, kunt u alleen webservices gebruiken die in hetzelfde internetdomein staan als het SWF-bestand dat de aanroep uitvoert, tenzij de server die als host fungeert voor de webservice ook de host is van een bestand met interdomeinbeleid dat toegang verleent aan aanvragen uit andere domeinen. Een techniek die vaak wordt gebruikt wanneer een bestand met interdomeinbeleid niet beschikbaar is, bestaat uit het indienen van aanvragen via uw eigen server middels een proxy. Adobe Blaze DS en Adobe LiveCycle bieden ondersteuning voor een dergelijke webserviceproxy.

Bij AIR-toepassingen is geen bestand met interdomeinbeleid vereist wanneer de webserviceaanroep komt uit de beveiligingssandbox van de toepassing. Inhoud van een AIR-toepassing wordt nooit aangeboden vanaf een extern domein. Daarom kan deze inhoud ook niet deelnemen aan de typen aanvallen waartegen bestanden met interdomeinbeleid beveiligen. Bij AIR-toepassingen die zijn gebaseerd op HTML kan de inhoud in de beveiligingssandbox van de toepassing interdomeinaanvragen indienen met XMLHttpRequests. U kunt inhoud in andere beveiligingssandboxen toestaan om interdomeinaanvragen met XMLHttpRequests in te dienen, op voorwaarde dat de inhoud wordt geladen in een iframe.

Meer Help-onderwerpen

"Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116

Adobe BlazeDS

Adobe LiveCycle ES2

REST-architectuur

XML-RPC

SOAP-protocol

REST-webserviceaanvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

REST-webservices gebruiken werkwoorden van een HTTP-methode om de basishandeling en URL-variabelen aan te geven en zo de details van de handeling te specificeren. Zo kan een aanvraag om gegevens op te halen voor een item gebruikmaken van het werkwoord GET en URL-variabelen om de naam van een methode en een item-id te specificeren. De resulterende URL-tekenreeks ziet er als volgt uit:

```
http://service.example.com/?method=getItem&id=d3452
```

Als u toegang wilt tot een REST-webservice met ActionScript, kunt u de klassen URLRequest, URLVariables en URLLoader gebruiken. In Javascript-code binnen een AIR-toepassing kunt u ook een XMLHttpRequest gebruiken.

Het programmeren van een REST-webserviceaanroep in ActionScript bestaat gewoonlijk uit de volgende stappen:

1. Maak een URLRequest-object.
2. Stel de service-URL en het werkwoord van de HTTP-methode in op het aanvraagobject.
3. Maak een URLVariables-object.
4. Stel de parameters van de serviceaanroep in als dynamische eigenschappen van het Variables-object.
5. Wijs het Variables-object toe aan de gegevenseigenschap van het aanvraagobject.
6. Verzend de aanroep naar de service met een URLLoader-object.
7. Verwerk de `complete`-gebeurtenis die wordt verzonden door de URLLoader die aangeeft dat de serviceaanroep is voltooid. Het is ook nuttig om te luisteren naar de verschillende foutgebeurtenissen die kunnen worden verzonden door een URLLoader-object.

Neem bijvoorbeeld een webservice die een testmethode beschikbaar maakt waarmee de parameters van de aanroep worden teruggekaatst naar de aanvrager. De service kan worden aangeroepen door de volgende ActionScript-code:

```
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 );
}
```

In Javascript binnen een AIR-toepassing kunt u dezelfde aanvraag uitvoeren met het XMLHttpRequest-object:

```
<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>
```

XML-RPC-webserviceaanvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij een XML-RPC-webservice hebben de aanroepparameters de vorm van een XML-document, en niet die van een set met URL-variabelen. Als u een transactie wilt uitvoeren met een XML-RPC-webservice, maakt u XML-bericht met een correcte indeling en verzendt u dit bericht naar de webservice met de HTTP `POST`-methode. Verder moet u de `Content-Type`-header configureren voor de aanvraag, zodat de server de aangevraagde gegevens behandelt als XML.

In het volgende voorbeeld wordt aangegeven hoe u dezelfde webserviceaanroep van het REST-voorbeeld kunt toepassen als een XML-RPC-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;
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 );
}
```

WebKit in AIR biedt geen ondersteuning voor E4X-syntaxis. De methode waarmee u het XML-document in het vorige voorbeeld hebt gemaakt, werkt dus niet in Javascript-code. In plaats hiervan moet u het XML-document maken met de DOM-methoden. U kunt het document ook maken als tekenreeks die u met de Javascript DOMParser-klasse converteert naar XML.

In het volgende voorbeeld worden met behulp van DOM-methoden een XML-RPC-bericht en een XMLHttpRequest gemaakt, waarmee de webservicetransactie wordt uitgevoerd:

```
<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>
```

SOAP-webserviceaanvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

SOAP bouwt verder op het algemene XML-RPC-webserviceconcept en biedt een rijkere, maar meer complexe methode om getypte gegevens over te dragen. SOAP-webservices bieden meestal een WSDL-bestand (Web Service Description Language) waarmee de webserviceaanroepen, gegevenstypen en service-URL's worden aangegeven. Alhoewel ActionScript 3 geen directe ondersteuning biedt voor SOAP, kunt u een SOAP-XML-bericht “met de hand” construeren en op de server posten, waarna u de resultaten kunt parseren. Voor de meeste SOAP-webservices (behalve de meest eenvoudige) bespaart u echter heel veel ontwikkelingstijd door een bestaande SOAP-bibliotheek te gebruiken.

Het Flex-raamwerk bevat bibliotheken voor toegang tot SOAP-webservices. In Flash Builder wordt de bibliotheek rpc.swc automatisch opgenomen in Flex-projecten, aangezien de bibliotheek onderdeel vormt van het Flex-raamwerk. In Flash Professional kunt u framework.swc en rpc.swc van Flex toevoegen aan het bibliotheekpad van een project, waarna u toegang krijgt tot de Flex-klassen met ActionScript.

Meer Help-onderwerpen

De Flex-webservicecomponent gebruiken in Flash Professional

Cristophe Coenraets: Real-time Trader Desktop voor Android

Een URL openen in een andere toepassing

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de functie `navigateToURL()` gebruiken om een URL te openen in een webbrowser of andere toepassing. Voor inhoud die in AIR wordt uitgevoerd, wordt de pagina in de standaardwebbrowser van het systeem geopend met de functie `navigateToURL()`.

Voor het URLRequest-object geeft u de parameter `request` van deze functie door. Alleen de eigenschap `url` wordt gebruikt.

De eerste parameter van de functie `navigateToURL()`, de parameter `navigate`, is een URLRequest-object (zie "De klasse URLRequest gebruiken" op pagina 858). De tweede parameter is een optionele `window`-parameter, waarin u de vensternaam kunt opgeven. De volgende code opent bijvoorbeeld de webpagina www.adobe.com:

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

Opmerking: *Wanneer u de functie `navigateToURL()` gebruikt, behandelt de runtime een URLRequest-object dat de methode POST gebruikt (een object waarvoor de eigenschap `method` is ingesteld op `URLRequestMethod.POST`) op dezelfde manier als het gebruik van de methode GET.*

Wanneer u de functie `navigateToURL()` gebruikt, zijn URI-schema's toegestaan op basis van de beveiligingssandbox van de code die de functie `navigateToURL()` aanroept.

Sommige API's stellen u in staat inhoud in een webbrowser te starten. Bepaalde URI-schema's zijn uit veiligheidsoverwegingen verboden wanneer deze API's in AIR worden gebruikt. De lijst van verboden schema's is afhankelijk van de beveiligingssandbox van de code die de API gebruikt. (Zie "Beveiliging in AIR" op pagina 1144 voor meer informatie over beveiligingssandboxen.)

Toepassingssandbox (alleen AIR)

Elk URI-schema kan worden gebruikt in een URL die wordt gestart door het uitvoeren van inhoud in de AIR-toepassingssandbox. Toepassingen moeten zijn geregistreerd om het URI-schema te kunnen afhandelen, anders levert het verzoek niets op. De volgende schema's worden op vele apparaten en computers ondersteund:

- `http:`
- `https:`
- `file:`
- `mailto:` - AIR leidt deze aanvragen naar de geregistreerde e-mailtoepassing van het systeem
- `sms:` — AIR stuurt `sms:`-verzoeken door aan de standaardtoepassing voor sms-berichten. De URL-indeling moet voldoen aan de gebruiksregels van het systeem waarop de toepassing wordt uitgevoerd. Op Android-systemen moet het URI-schema bijvoorbeeld in kleine letters worden ingevoerd.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- `tel:` — AIR stuurt `tel:`-verzoeken door aan de standaardtelefoontoepassing. De URL-indeling moet voldoen aan de gebruiksregels van het systeem waarop de toepassing wordt uitgevoerd. Op Android-systemen moet het URI-schema bijvoorbeeld in kleine letters worden ingevoerd.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- `market:` — AIR stuurt `market:`-verzoeken door aan de Market-toepassing die doorgaans wordt ondersteund op Android-apparaten.

```
navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
```

Toepassingen kunnen aangepaste URI-schema's definiëren en registreren, indien het besturingssysteem dat toestaat. U kunt een URL maken met gebruik van het schema en de toepassing starten vanuit AIR.

Externe sandboxen

De volgende schema's zijn toegestaan. Gebruik deze schema's zoals u ze zou gebruiken in een webbrowser.

- `http:`
- `https:`
- `mailto:` - AIR leidt deze aanvragen naar de geregistreerde e-mailtoepassing van het systeem

Alle andere URI-schema's zijn verboden.

Lokaal-met-bestandssysteem sandbox

De volgende schema's zijn toegestaan. Gebruik deze schema's zoals u ze zou gebruiken in een webbrowser.

- `file:`
- `mailto:` - AIR leidt deze aanvragen naar de geregistreerde e-mailtoepassing van het systeem

Alle andere URI-schema's zijn verboden.

Lokaal-met-netwerk sandbox

De volgende schema's zijn toegestaan. Gebruik deze schema's zoals u ze zou gebruiken in een webbrowser.

- `http:`
- `https:`
- `mailto:` - AIR leidt deze aanvragen naar de geregistreerde e-mailtoepassing van het systeem

Alle andere URI-schema's zijn verboden.

Lokaal-vertrouwde sandbox

De volgende schema's zijn toegestaan. Gebruik deze schema's zoals u ze zou gebruiken in een webbrowser.

- `file:`
- `http:`
- https:
- `mailto:` - AIR leidt deze aanvragen naar de geregistreerde e-mailtoepassing van het systeem

Alle andere URI-schema's zijn verboden.

Hoofdstuk 45: Communiceren met andere instanties van Flash Player en AIR

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse LocalConnection maakt communicatie mogelijk tussen Adobe® AIR®-toepassingen en tussen SWF-inhoud die in de browser wordt uitgevoerd. U kunt de klasse LocalConnection ook gebruiken om te communiceren tussen een AIR-toepassing en SWF-inhoud in de browser. Met de klasse LocalConnection kunt u zeer veelzijdige toepassingen maken die gegevens kunnen delen tussen instanties van Flash Player en AIR.

Informatie over de klasse LocalConnection

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse LocalConnection biedt u de mogelijkheid SWF-bestanden te ontwikkelen die instructies kunnen verzenden naar andere SWF-bestanden zonder gebruik te maken van de methode fscommand() of JavaScript. Een object LocalConnection kan alleen communiceren via SWF-bestanden die op dezelfde clientcomputer worden uitgevoerd, maar een object LocalConnection kan door verschillende toepassingen worden uitgevoerd. Zo kunnen een SWF-bestand dat wordt uitgevoerd in een browser en een SWF-bestand dat wordt uitgevoerd in een projector informatie delen, waarbij de projector lokale informatie bijhoudt en het SWF-bestand in de browser extern verbinding maakt. (Een projector is een SWF-bestand dat is opgeslagen in een indeling die als een zelfstandige toepassing kan worden uitgevoerd. Met andere woorden, de projector vereist niet dat Flash Player is geïnstalleerd omdat dit programma is ingesloten in het uitvoerbare bestand.)

Een object LocalConnection kan worden gebruikt om te communiceren tussen SWF-bestanden die verschillende versies van ActionScript gebruiken:

- Een object LocalConnection uit ActionScript 3.0 kan communiceren met objecten LocalConnection die in ActionScript 1.0 of 2.0 zijn gemaakt.
- Een object LocalConnection uit ActionScript 1.0 of 2.0 kan communiceren met objecten LocalConnection die in ActionScript 3.0 zijn gemaakt.

Flash Player handelt deze communicatie tussen objecten LocalConnection van verschillende versies automatisch af.

De eenvoudigste manier om een LocalConnection-object te gebruiken, is om alleen communicatie tussen LocalConnection-objecten in hetzelfde domein of dezelfde AIR-toepassing toe te staan. Zo hoeft u zich geen zorgen te maken over de beveiliging. Wanneer het echter noodzakelijk is communicatie tussen domeinen toe te staan, kunt u op verschillende manieren beveiligingsmaatregelen implementeren. Zie de discussie over de parameter `connectionName` van de methode `send()` en de vermeldingen `allowDomain()` en `domain` in de vermelding van de klasse LocalConnection in de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie.

U kunt objecten LocalConnection gebruiken voor het verzenden en ontvangen van gegevens binnen één SWF-bestand, maar Adobe beveelt dit niet aan. Gebruik in plaats hiervan gezamenlijke objecten.

U kunt op drie manieren callback-methoden toevoegen aan uw objecten LocalConnection:

- Maak een subklasse van de klasse LocalConnection en voeg methoden toe.

- Stel de eigenschap `LocalConnection.client` in op een object dat de methoden implementeert.
- Maak een dynamische klasse die LocalConnection uitbreidt en verbind de methoden dynamisch.

U kunt ten eerste callback-methoden toevoegen door de klasse LocalConnection uit te breiden. U definieert de methoden binnen de aangepaste klasse in plaats van ze dynamisch toe te voegen aan de instantie van LocalConnection. Dit wordt geïllustreerd in de volgende code:

```
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);
        }
    }
}
```

Om een nieuwe instantie van de klasse CustomLocalConnection te maken, kunt u de volgende code gebruiken:

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

U kunt ten tweede ook callback-methoden toevoegen met de eigenschap `LocalConnection.client`. Hierbij maakt u een aangepaste klasse en wijst u een nieuwe instantie toe aan de eigenschap `client`, zoals in de volgende code wordt getoond:

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

De eigenschap `LocalConnection.client` geeft de callback-methoden aan die moeten worden aangeroepen. In de voorgaande code werd de eigenschap `client` ingesteld op een nieuwe instantie van een aangepaste klasse, CustomClient. De standaardwaarde voor de eigenschap `client` is de huidige instantie van LocalConnection. U kunt de eigenschap `client` gebruiken als u twee gegevenshandlers gebruikt die dezelfde reeks methoden hebben, maar anders werken, bijvoorbeeld in een toepassing waar een knop in het ene venster de weergave in een tweede venster wijzigt.

U kunt de klasse CustomClient maken door de volgende code te gebruiken:

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

De derde manier om callback-methoden toe te voegen, een dynamische klasse maken en de methoden dynamisch koppelen, lijkt op het gebruik van de klasse LocalConnection in eerdere versies van ActionScript, zoals u ziet in de volgende code:

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

Callback-methoden kunnen dynamisch aan deze klasse worden toegevoegd met de volgende code:

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

De voorgaande methode om callback-methoden toe te voegen wordt niet aanbevolen omdat de code niet erg overdraagbaar is. Als u deze methode om lokale verbindingen te maken gebruikt, zou dit bovendien kunnen leiden tot prestatievermindering omdat de toegang tot dynamische eigenschappen veel trager is dan de toegang tot verzegelde eigenschappen.

isPerUser-eigenschap

De `isPerUser`-eigenschap is aan Flash Player (10.0.32) en AIR (1.5.2) toegevoegd om een conflict op te lossen dat optreedt wanneer meerdere gebruikers zijn aangemeld bij een Mac-computer. Op andere besturingssystemen wordt deze eigenschap genegeerd, aangezien de lokale verbinding altijd al bestemd was voor individuele gebruikers. In nieuwe code dient de `isPerUser`-eigenschap te worden ingesteld op `true`. De standaardwaarde is momenteel echter `false` ten behoeve van compatibiliteit met oudere versies. Het is mogelijk dat deze standaardinstelling in nieuwe versies van de runtimes wordt veranderd.

Berichten versturen tussen twee toepassingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de LocalConnection-klasse kunt u communiceren tussen verschillende AIR-toepassingen en tussen verschillende Adobe® Flash® Player (SWF)-toepassingen die in een browser actief zijn. U kunt de klasse LocalConnection ook gebruiken om te communiceren tussen een AIR-toepassing en SWF-toepassing in een browser.

U zou bijvoorbeeld meerdere instanties van Flash Player aan een webpagina kunnen toevoegen of een Flash Player-instantie gegevens laten ophalen uit een Flash Player-instantie in een pop-upvenster.

De volgende code definieert een LocalConnection-object dat als server fungeert en binnenkomende LocalConnection-oproepen van andere toepassingen accepteert:

```
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");
            }
        }
    }
}
```

Deze code maakt eerst een LocalConnection-object met de naam `lc` en stelt de eigenschap `client` in op een `clientObject`-object. Als een andere toepassing een methode in deze LocalConnection-instantie aanroept, zoekt de runtime naar die methode in het `clientObject`-object.

Als er al een verbinding met de opgegeven naam bestaat, wordt de uitzondering ArgumentError gegenereerd om aan te geven dat de verbindingspoging is mislukt omdat het object al is verbonden.

Wanneer een instantie van Flash Player verbinding maakt met dit SWF-bestand en probeert een methode aan te roepen voor de opgegeven lokale verbinding, wordt de aanvraag verzonden naar de klasse die is opgegeven in de eigenschap `client`, die is ingesteld op de klasse 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");
        }
    }
}
```

Als u een LocalConnection-server wilt maken, roept u de methode `LocalConnection.connect()` aan en geeft u een unieke naam voor de verbinding op. Als er al een verbinding is met de opgegeven naam, wordt de fout ArgumentError gegenereerd om aan te geven dat de verbindingspoging is mislukt omdat het object al is verbonden.

In het volgende fragment ziet u hoe u een nieuwe LocalConnection maakt met de naam `conn1`:

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

Als u verbinding wilt maken met de primaire toepassing vanuit een secundaire toepassing, moet u eerst een LocalConnection-object maken in het verzendende LocalConnection-object. Roep vervolgens de methode `LocalConnection.send()` aan met de naam van de verbinding en de naam van de methode die moeten worden uitgevoerd. Stuur bijvoorbeeld de `doQuit`-methode naar het LocalConnection-object dat u eerder hebt gemaakt, met behulp van de volgende code:

```
sendingConnection.send("conn1", "doQuit");
```

Deze code maakt verbinding met een bestaand LocalConnection-object met de verbindingsnaam `conn1` en roept de methode `doMessage()` in de externe toepassing op. Als u parameters naar de externe toepassing wilt verzenden, geeft u aanvullende argumenten op na de methodenaam in de methode `send()`, zoals in het volgende fragment:

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

Verbinding maken met inhoud in verschillende domeinen en met AIR-toepassingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u alleen communicatie vanuit specifieke domeinen wenst toe te staan, roept u de methode `allowDomain()` of `allowInsecureDomain()` van de klasse LocalConnection op en geeft u een lijst met een of meer domeinen door die toegang mogen hebben tot dit LocalConnection-object.

In eerdere versies van ActionScript waren `LocalConnection.allowDomain()` en `LocalConnection.allowInsecureDomain()` callback-methoden die door ontwikkelaars moesten worden geïmplementeerd en een Booleaanse waarde moesten retourneren. In ActionScript 3.0 zijn `LocalConnection.allowDomain()` en `LocalConnection.allowInsecureDomain()` beide ingebouwde methoden die ontwikkelaars kunnen aanroepen op dezelfde manier als `Security.allowDomain()` en `Security.allowInsecureDomain()`, waarbij ze een of meer namen van toegestane domeinen doorgeven.

In Flash Player 8 zijn voor het eerst beveiligingsbeperkingen voor lokale SWF-bestanden opgenomen. Een SWF-bestand dat toegang heeft tot internet, kan niet ook toegang tot het lokale bestandssysteem hebben. Wanneer u `localhost` opgeeft, heeft elk lokaal SWF-bestand toegang tot dit SWF-bestand. Als de methode `LocalConnection.send()` probeert te communiceren met een SWF-bestand uit een beveiligingssandbox waartoe de aanroepende code geen toegang heeft, wordt de gebeurtenis `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) verzonden. U kunt deze fout omzeilen door het domein van de oproeper op te geven in de methode `LocalConnection.allowDomain()` van de ontvanger.

Er zijn twee speciale waarden die u kunt doorgeven aan de methoden `LocalConnection.allowDomain()` en `LocalConnection.allowInsecureDomain()`: `*` en `localhost`. Het sterretje `(*)` staat toegang toe vanuit alle domeinen. Met de tekenreeks `localhost` zijn oproepen van de toepassing mogelijk vanuit inhoud die lokaal is geïnstalleerd maar zich buiten de resourcemap van de toepassing bevindt.

Als de methode `LocalConnection.send()` probeert te communiceren met een toepassing vanuit een beveiligingssandbox waartoe de oproepende code geen toegang heeft, wordt de gebeurtenis `securityError` (`SecurityErrorEvent.SECURITY_ERROR`) verzonden. U kunt deze fout omzeilen door het domein van de oproeper op te geven in de methode `LocalConnection.allowDomain()` van de ontvanger.

Als u alleen communicatie tussen inhoud in hetzelfde domein implementeert, kunt u de parameter `connectionName` opgeven die niet met een onderstrepingsteken (_) begint en die geen domeinnaam opgeeft (bijvoorbeeld `myDomain:connectionName`). Gebruik dezelfde tekenreeks in de opdracht `LocalConnection.connect(connectionName)`.

Als u communicatie tussen inhoud in verschillende domeinen implementeert, geeft u voor de parameter `connectionName` een waarde op die begint met een onderstrepingsteken. Als u het onderstrepingsteken gebruikt, kunt u de inhoud met het ontvangende LocalConnection-object gemakkelijker overdragen tussen domeinen. Hier volgen twee mogelijke scenario's:

- Wanneer de tekenreeks voor `connectionName` niet met een onderstrepingsteken begint, voegt de runtime aan de superdomeinnaam een voorvoegsel plus dubbele punt toe (bijvoorbeeld `myDomain:connectionName`). Hoewel dit ervoor zorgt dat er geen conflicten optreden tussen de verbinding en andere verbindingen met dezelfde naam in een ander domein, moet elk verzendend LocalConnection-object dit superdomein opgeven (bijvoorbeeld `myDomain:connectionName`). Als u het HTML- of SWF-bestand met het ontvangende LocalConnection-object naar een ander domein verplaatst, wijzigt de runtime het voorvoegsel, zodat dit het nieuwe superdomein weerspiegelt (bijvoorbeeld `anotherDomain:connectionName`). Alle verzendende LocalConnection-objecten moeten handmatig worden bewerkt, zodat ze naar het nieuwe superdomein wijzen.
- Wanneer de tekenreeks voor `connectionName` met een onderstrepingsteken begint (bijvoorbeeld `_connectionName"`), voegt de runtime geen voorvoegsel toe aan de tekenreeks. Dit houdt in dat de ontvangende en verzendende LocalConnection-objecten identieke tekenreeksen gebruiken voor `connectionName`. Als het ontvangende object door middel van `LocalConnection.allowDomain()` opgeeft dat verbindingen van elk domein worden geaccepteerd, kunt u het HTML- of SWF-bestand met het ontvangende LocalConnection-object naar een ander domein verplaatsen zonder dat u verzendende LocalConnection-objecten hoeft te wijzigen.

  Een nadeel van namen met onderstrepingsteken in `connectionName` is de kans op botsingen, bijvoorbeeld als twee toepassingen verbinding proberen te maken met dezelfde `connectionName`. Een tweede, gerelateerd nadeel betreft de beveiliging. Verbindingsnamen die gebruikmaken van een onderstrepingsteken identificeren niet het domein van de luisterende toepassing. Daarom genieten domeinspecifieke namen de voorkeur.

Adobe AIR

Om te kunnen communiceren met inhoud die in de beveiligingssandbox van de AIR-toepassing wordt uitgevoerd (inhoud die samen met de AIR-toepassing is geïnstalleerd), dient u een voorvoegsel aan de verbindingsnaam toe te voegen waarin het superdomein dat de AIR-toepassing identificeert tot uitdrukking komt. De superdomeinreeks begint met `app#`, gevolgd door achtereenvolgens de toepassings-id, een punt (.) en de uitgevers-id (als deze gedefinieerd is). Zo is het juiste superdomein voor gebruik in de `connectionName`-parameter voor een toepassing met de toepassings-id `com.example.air.MyApp`, maar zonder uitgevers-id: `"app#com.example.air.MyApp"`. Als de basisnaam van de verbinding "appConnection" is, dient u de volgende volledige reeks te gebruiken in de `connectionName`-parameter: `"app#com.example.air.MyApp:appConnection"`. Als de toepassing de uitgevers-id heeft, moet die id ook worden opgenomen in de superdomeinreeks: `"app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1"`.

Als u toestaat dat een andere AIR-toepassing via de lokale verbinding met uw toepassing communiceert, dient u de waarde `allowDomain()` van het LocalConnection-object aan te roepen en de domeinnaam van de lokale verbinding door te geven. Voor een AIR-toepassing wordt deze domeinnaam gevormd op basis van de toepassings- en uitgevers-id, net zoals bij de verbindingstekenreeks. Voorbeeld: als de verzendende AIR-toepassing de toepassings-id `com.example.air.FriendlyApp` en de uitgevers-id `214649436BD677B62C33D02233043EA236D13934.1`, heeft, gebruikt u de volgende domeintekenreeks om deze toepassing toestemming te geven om te verbinden: `app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1`. (Met ingang van AIR 1.5.3 beschikken niet alle AIR-toepassingen over gebruikers-id's.)

Hoofdstuk 46: Communiceren met native processen in AIR

Adobe AIR 2 of hoger

Vanaf Adobe AIR 2 kunnen AIR-toepassingen via de opdrachtregel andere native processen uitvoeren en hiermee communiceren. Een AIR-toepassing kan bijvoorbeeld via de standaardinvoer- en uitvoerstreams een proces uitvoeren en hiermee communiceren.

Voor communicatie met een native proces moet u van een AIR-toepassing een pakket maken dat via een native installatieprogramma kan worden geïnstalleerd. Het bestandstype van het native installatieprogramma is speciaal ontworpen voor het besturingssysteem waarvoor het is gemaakt:

- Bij Mac OS-systemen is dit een DMG-bestand.
- Bij Windows-systemen is dit een EXE-bestand.
- Bij Linux-systemen is dit een RPM- of DEB-pakket.

Deze toepassingen zijn ook wel bekend als toepassingen in het uitgebreide bureaubladprofiel. U kunt een native installatiebestand maken door de optie `-target native` op te geven, wanneer u met ADT de opdracht `-package` aanroept.

Meer Help-onderwerpen

flash.filesystem.File.openWithDefaultApplication()

flash.desktop.NativeProcess

Overzicht van communicatie met native processen

Adobe AIR 2 of hoger

Een AIR-toepassing in het uitgebreide bureaubladprofiel kan een bestand uitvoeren, net alsof het uitvoerbare bestand wordt aangeroepen door de opdrachtregel. De toepassing kan communiceren met de standaardstreams van het native proces. Standaardstreams zijn onder meer de standaardinvoerstream (stdin), de standaarduitvoerstream (stdout) en de standaardfoutenstream (stderr).

Opmerking: *Toepassingen in het uitgebreide bureaubladprofiel kunnen ook bestanden en toepassingen starten met de methode* `File.openWithDefaultApplication()`*. Door het gebruik van deze methode heeft de AIR-toepassing echter geen toegang tot de standaardstreams. Zie "Bestanden openen met de standaardsysteemtoepassing" op pagina 719 voor meer informatie.*

Het volgende codevoorbeeld laat zien hoe de toepassing test.exe in de toepassingsmap moet worden gestart: De toepassing geeft het argument `"hello"` door als een opdrachtregelargument en er wordt een gebeurtenislistener toegevoegd aan de standaarduitvoerstream van het proces:

```
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);
}
```

Een native proces starten en sluiten

Adobe AIR 2 of hoger

Als u een native proces wilt starten, stelt u een NativeProcessInfo-object als volgt in:

- Maak een verwijzing naar het bestand dat u wilt starten.
- Sla opdrachtregelargumenten op die moeten worden doorgegeven aan het proces wanneer dit wordt gestart. (optioneel)
- Stel de werkmap van het proces in (optioneel)

Start het native proces door het NativeProcessInfo-object door te geven als de parameter van de methode `start()` van een NativeProcess-object.

De volgende code laat bijvoorbeeld zien hoe de toepassing test.exe in de toepassingsmap moet worden gestart: De toepassing geeft het argument `"hello"` door en stelt de documentenmap van de gebruiker in als de werkmap:

```
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);
```

Als u het proces wilt beëindigen, roept u de methode `exit()` van het NativeProcess-object aan.

Als u een uitvoerbaar bestand wilt opnemen in de geïnstalleerde toepassing, dient u in het bestandsysteem te controleren of het bestand uitvoerbaar is wanneer u een pakket maakt van de toepassing. (Indien noodzakelijk, kunt u op Mac en Linux chmod gebruiken om de uitvoerbare markering in te stellen.)

Communiceren met een native proces

Adobe AIR 2 of hoger

Als een AIR-toepassing eenmaal een native proces heeft gestart, kan de toepassing communiceren met de standaardinvoerstreams, de standaarduitvoerstreams en de standaardfoutenstreams van het proces.

Met de volgende eigenschappen van het object NativeProcess kunt u streams lezen en gegevens naar streams schrijven:

- `standardInput` - Toegang tot de gegevens van de standaardinvoerstream.
- `standardOutput` - Toegang tot de gegevens van de standaarduitvoerstream.
- `standardError` - Toegang tot de gegevens van de standaardfoutenstream.

Schrijven naar de standaardinvoerstream

U kunt gegevens naar de standaardinvoerstream schrijven met de schrijfmethoden van de `standardInput`-eigenschap van het NativeProcess-object. Als de AIR-toepassing gegevens naar het proces schrijft, verzendt het NativeProcess-object `standardInputProgress`-gebeurtenissen.

Als er zich bij het schrijven naar de standaardinvoerstream een fout voordoet, verzendt het object NativeProcess een `ioErrorStandardInput`-gebeurtenis.

U kunt de invoerstream sluiten door de methode `closeInput()` van het object NativeProcess aan te roepen. Bij het sluiten van de invoerstream verzendt het NativeProcess-object een `standardInputClose`-gebeurtenis.

```
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();
}
```

De standaarduitvoerstream lezen

U kunt gegevens in de standaarduitvoerstream lezen met de leesmethoden van deze eigenschap. Wanneer de AIR-toepassing uitvoerstreamgegevens uit het proces ophaalt, verzendt het NativeProcess-object `standardOutputData`-gebeurtenissen.

Als er zich bij het schrijven naar de standaarduitvoerstream een fout voordoet, verzendt het object NativeProcess een `standardOutputError`-gebeurtenis.

Als de uitvoerstream door het proces wordt gesloten, verzendt het object NativeProcess een `standardOutputClose`-gebeurtenis.

Bij het lezen van gegevens van de standaardinvoerstream moeten de gegevens worden gelezen op het moment dat ze worden gegenereerd. Met andere woorden: u kunt het beste een gebeurtenislistener toevoegen voor de `standardOutputData`-gebeurtenis. In de `standardOutputData`-gebeurtenislistener leest u dan de gegevens van de `standardOutput`-eigenschap van het NativeProcess-object. Wacht niet op de `standardOutputClose`-gebeurtenis of de `exit`-gebeurtenis voordat u alle gegevens gaat lezen. Als u de gegevens niet leest op het moment dat ze door het native proces worden gegenereerd, kan de buffer vol raken, en kunnen er gegevens verdwijnen. Door een volle buffer kan het native proces stagneren bij het schrijven van meer gegevens. Als u geen gebeurtenislistener registreert voor de `standardOutputData`-gebeurtenis, raakt de buffer niet vol en stagneert het proces niet. In dit geval hebt u geen toegang tot de gegevens.

```
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);
}
```

De standaardfoutenstream lezen

U kunt gegevens in de standaardfoutenstream lezen met de leesmethoden van deze eigenschap. Wanneer de AIR-toepassing gegevens in de foutenstream leest, verzendt het NativeProcess-object `standardErrorData`-gebeurtenissen.

Als er zich bij het schrijven naar de standaardfoutenstream een fout voordoet, verzendt het object NativeProcess een `standardErrorIoError`-gebeurtenis.

Als de foutenstream door het proces wordt gesloten, verzendt het object NativeProcess een `standardErrorClose`-gebeurtenis.

Bij het lezen van gegevens van de standaardfoutstream moeten de gegevens worden gelezen op het moment dat ze worden gegenereerd. Met andere woorden: u kunt het beste een gebeurtenislistener toevoegen voor de `standardErrorData`-gebeurtenis. In de `standardErrorData`-gebeurtenislistener leest u dan de gegevens van de `standardError`-eigenschap van het NativeProcess-object. Wacht niet op de `standardErrorClose`-gebeurtenis of de `exit`-gebeurtenis voordat u alle gegevens gaat lezen. Als u de gegevens niet leest op het moment dat ze door het native proces worden gegenereerd, kan de buffer vol raken, en kunnen er gegevens verdwijnen. Door een volle buffer kan het native proces stagneren bij het schrijven van meer gegevens. Als u geen gebeurtenislistener registreert voor de `standardErrorData`-gebeurtenis, raakt de buffer niet vol en stagneert het proces niet. In dit geval hebt u geen toegang tot de gegevens.

```
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);
}
```

Veiligheidsoverwegingen voor communicatie met native processen

Adobe AIR 2 of hoger

De API van het native proces kan elk uitvoerbaar bestand op het systeem van de gebruiker uitvoeren. Wees zeer voorzichtig bij het opstellen en uitvoeren van opdrachten. Als een deel van een uit te voeren opdracht van een externe bron komt, moet u zorgvuldig controleren of de opdracht veilig is om uit te voeren. Ook dient de AIR-toepassing alle gegevens die worden doorgegeven aan een uitvoerend proces te valideren.

Het valideren van invoer kan echter moeilijk zijn. Om zulke moeilijkheden te vermijden, kunt u het best een native toepassing (zoals een EXE-bestand in Windows) met specifieke API's te schrijven. Deze API's verwerken dan alleen de opdrachten die zijn gedefinieerd door de toepassing. De toepassing accepteert dan bijvoorbeeld slechts een beperkt aantal instructies via de standaardinvoerstream.

AIR op Windows staat u niet toe om .bat-bestanden rechtstreeks uit te voeren. BAT-bestanden in Windows worden uitgevoerd door het CMD.EXE, een toepassing waarmee opdrachten worden geïnterpreteerd. Wanneer u dit .bat-bestand oproept, kan deze opdrachttoepassing argumenten verwerken die aan de opdracht doorgeeft om aanvullende toepassingen te starten. Een schadelijke invoeging van extra tekens in de argumenttekenreeks kan ertoe leiden dat cmd.exe een schadelijke of onveilige toepassing uitvoert. Zonder de juiste gegevensvalidatie kan uw AIR-toepassing bijvoorbeeld `myBat.bat myArguments c:/evil.exe` misschien oproepen. De opdrachttoepassing start de evil.exe-toepassing naast het uitvoeren van uw batchbestand.

Hoofdstuk 47: De externe API gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de externe API van ActionScript 3.0 (flash.external.ExternalInterface) is directe communicatie mogelijk tussen ActionScript en de containertoepassing waarin Adobe Flash Player wordt uitgevoerd. Gebruik de ExternalInterface-API om interactie tussen een SWF-document en JavaScript op een HTML-pagina tot stand te brengen.

U kunt de externe API gebruiken om te communiceren met een containertoepassing en gegevens door te geven tussen ActionScript en JavaScript op een HTML-pagina.

Sommige algemene externe API-taken zijn:

- Informatie ophalen over de containertoepassing.
- ActionScript gebruiken om code aan te roepen op een webpagina die wordt weergegeven in een browser of een AIR-bureaubladtoepassing.
- ActionScript-code aanroepen van een webpagina.
- Een proxy maken om het aanroepen van ActionScript-codes uit een webpagina te vereenvoudigen.

Opmerking: *Deze discussie over de externe interface beslaat alleen de communicatie tussen ActionScript in een SWF-bestand en de containertoepassing die een verwijzing naar Flash Player of de instantie bevat waarin het SWF-bestand is geladen. Elk ander gebruik van Flash Player binnen een toepassing valt buiten het bereik van deze documentatie. Flash Player is bedoeld om als browserinsteekmodule of als projector (zelfstandige toepassing) te worden gebruikt. De ondersteuning voor andere gebruiksscenario's is mogelijk beperkt.*

De externe API in AIR gebruiken

Aangezien een AIR-toepassing niet beschikt over een externe container, wordt deze externe interface niet algemeen toegepast en is de interface over het algemeen niet nodig. Wanneer uw AIR-toepassing rechtstreeks een SWF-bestand laadt, kan de toepassingscode rechtstreeks communiceren met de ActionScript-code in het SWF-bestand (afhankelijk van beperkingen met betrekking tot de beveiligingssandbox).

Wanneer uw AIR-toepassing echter een SWF-bestand laadt met een HTML-pagina in een HTMLLoader-object (of een HTML-component in Flex), fungeert het HTMLLoader-object als de externe container. Zo kunt u met de externe interface communiceren tussen de ActionScript-code in het geladen SWF-bestand en de JavaScript-code in de HTML-pagina die geladen is in de HTMLLoader.

Basisbeginselen van de externe API

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Hoewel een SWF-bestand in sommige gevallen apart kan worden uitgevoerd (bijvoorbeeld, als u Adobe® Flash® Professional gebruikt om een SWF-projector te maken), wordt een SWF-toepassing meestal uitgevoerd als een element binnen een andere toepassing. Normaal gesproken is de container die het SWF-bestand bevat, een HTML-bestand; wat minder vaak wordt een SWF-bestand gebruikt voor de complete of een deel van de gebruikersinterface van een bureaubladtoepassing.

Wanneer u werkt met meer geavanceerde toepassingen, kan het nodig zijn communicatie tussen het SWF-bestand en de containertoepassing tot stand te brengen. Het is bijvoorbeeld gebruikelijk dat een webpagina tekst of andere informatie in HTML weergeeft en een SWF-bestand bevat om dynamische visuele inhoud, zoals diagrammen of videobeelden, weer te geven. In dat geval wilt u bijvoorbeeld dat een SWF-bestand wordt gewijzigd, wanneer de gebruiker op een knop in de webpagina klikt. ActionScript bevat een externe API, een mechanisme dat deze vorm van communicatie tussen ActionScript in een SWF-bestand en een andere code in de containertoepassing mogelijk maakt.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot deze functie:

Containertoepassing De toepassing van waaruit Flash Player een SWF-bestand uitvoert, zoals een webbrowser en HTML-pagina met Flash Player-inhoud of een AIR-toepassing die het SWF-bestand in een webpagina laadt.

Projector Een uitvoerbaar bestand dat SWF-inhoud en een ingesloten versie van Flash Player bevat. U kunt een projectorbestand maken met Flash Professional of met de zelfstandige Flash Player. Projectors worden normaal gesproken gebruikt voor het distribueren van SWF-bestanden via cd-rom's, of in vergelijkbare situaties waarbij de downloadgrootte geen bezwaar is en de auteur van het SWF-bestand er zeker van wil zijn dat de gebruiker het SWF-bestand kan uitvoeren, ongeacht of de Flash Player op de computer van de gebruiker is geïnstalleerd.

Proxy Een tussenliggende toepassing of code die een code binnen een toepassing (de externe toepassing) aanroept voor een andere toepassing (de aanroepende toepassing) en waarden retourneert aan de aanroepende toepassing. Een proxy kan om verschillende redenen worden gebruikt, bijvoorbeeld:

- Om het proces waarbij externe functieaanroepen worden uitgevoerd te vereenvoudigen, door native functieaanroepen in de aanroepende toepassing om te zetten in de indeling die door de externe toepassing wordt begrepen..
- Voor het omzeilen van beveiligingsrestricties en andere beperkingen, die ervoor zorgen dat de aanroeper niet rechtstreeks kan communiceren met de externe toepassing.

Serialiseren Het omzetten van objecten of gegevenswaarden in een indeling die gebruikt kan worden om waarden in berichten uit te wisselen tussen twee programmeersystemen, bijvoorbeeld via internet of tussen twee verschillende toepassingen die op één computer worden uitgevoerd.

Werken met voorbeelden

Veel van de voorziene codevoorbeelden zijn kleine lijsten codes voor demonstratiedoeleinden en geen volledig werkende voorbeelden of code die waarden controleert. Omdat het gebruik van een externe API (per definitie) het schrijven van een ActionScript-code en een code in een containertoepassing vereist, moet u bij het testen van een voorbeeld een container maken (bijvoorbeeld een internetpagina met een SWF-bestand) en de codelijsten gebruiken om met de container te communiceren.

U kunt als volgt een voorbeeld van communicatie tussen JavaScript en ActionScript testen:

1. Maak een nieuw document met Flash Professional en sla het op uw computer op.
2. Kies vanuit het hoofdmenu Bestand > Publicatie-instellingen.
3. Controleer in het tabblad Indelingen in het dialoogvenster Publicatie-instellingen, of de selectievakjes Flash en HTML zijn ingeschakeld.
4. Klik op de knop Publiceren. Hierdoor worden in dezelfde map een SWF- en een HTML-bestand gemaakt met de naam waaronder u het document hebt opgeslagen. Klik op OK om het dialoogvenster Publicatie-instellingen te sluiten.

5 Schakel het selectievakje HTML uit. Nadat de HTML-pagina is gemaakt, wijzigt u deze door hieraan de juiste JavaScript-code toe te voegen. Wanneer u het selectievakje HTML uitschakelt, zorgt u dat Flash, nadat u de HTML-pagina hebt gewijzigd, uw wijzigingen bij publicatie van het SWF-bestand niet overschrijft met een nieuwe HTML-pagina.

6 Klik op OK om het dialoogvenster Publicatie-instellingen te sluiten.

7 Open met een HTML- of tekstbewerkingstoepassing het HTML-bestand dat met Flash is gemaakt wanneer u het SWF-bestand publiceert. Voeg in de HTML-broncode begin- en eind`script`tags toe en kopieer hierin de JavaScript-code uit de voorbeeldcode:

```
<script>
// add the sample JavaScript code here
</script>
```

8 Sla het HTML-bestand op en ga terug naar Flash.

9 Selecteer het hoofdframe in Frame 1 van de tijdlijn en open het deelvenster Handelingen.

10 Kopieer het ActionScript-codevoorbeeld in het Script-veld.

11 Kies vanuit het hoofdmenu Bestand > Publiceren, om het SWF-bestand bij te werken met de wijzigingen.

12 Open met een webbrowser de HTML-pagina die u bewerkt hebt, om de pagina- en testcommunicatie tussen ActionScript en de HTML-pagina weer te geven.

U kunt als volgt een voorbeeld van communicatie tussen ActionScript en een ActiveX-container testen:

1 Maak een nieuw document met Flash Professional en sla het op uw computer op. Mogelijk wilt u dit opslaan in de map waarin uw containertoepassing verwacht dat het SWF-bestand zich bevindt.

2 Kies vanuit het hoofdmenu Bestand > Publicatie-instellingen.

3 Controleer in het tabblad Indelingen in het dialoogvenster Publicatie-instellingen, of het selectievakje Flash is ingeschakeld.

4 Klik in het veld Bestand naast het selectievakje Flash op het mappictogram om de map te selecteren waarin u het SWF-bestand wilt publiceren. Door de locatie van uw SWF-bestand in te stellen, kunt u (bijvoorbeeld) het document in een map bewaren, maar plaatst u het gepubliceerde SWF-bestand in een andere map, zoals de map met de broncode voor de containertoepassing.

5 Selecteer het hoofdframe in Frame 1 van de tijdlijn en open het deelvenster Handelingen.

6 Kopieer de ActionScript-code voor het voorbeeld in het Script-veld.

7 Kies vanuit het hoofdmenu Bestand > Publiceren om het SWF-bestand opnieuw te publiceren.

8 Maak uw containertoepassing en voer deze uit om de communicatie tussen ActionScript en de containertoepassing te testen.

Raadpleeg het volgende onderwerp voor volledige voorbeelden van het gebruik van de externe API om te communiceren met een HTML-pagina:

- “Voorbeeld van externe API: communicatie tussen ActionScript en JavaScript in een webbrowser” op pagina 897

Deze voorbeelden bevatten de volledige code, waaronder ActionScript en de foutcontrolecode voor de container, die u moet gebruiken wanneer u code schrijft met de externe API. Zie het voorbeeld voor de klasse ExternalInterface in de ActionScript 3.0 Reference voor een ander volledig voorbeeld van het gebruik van de externe API.

Vereisten en voordelen van de externe API

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De externe API vormt het deel van ActionScript dat communicatie mogelijk maakt tussen ActionScript en code en wordt uitgevoerd binnen een 'externe toepassing' die dient als container voor Flash Player (normaal gesproken een webbrowser of zelfstandige projectortoepassing). In ActionScript 3.0 wordt de functionaliteit van de externe API verzorgd door de klasse ExternalInterface. In eerdere versies dan Flash Player 8 werd de handeling `fscommand()` gebruikt om communicatie met de containertoepassing tot stand te brengen. De ExternalInterface-klasse is een vervanging voor `fscommand()`.

Opmerking: *Als u bent aangewezen op de oude functie `fscommand()`, bijvoorbeeld voor compatibiliteit met oudere toepassingen of om te communiceren met een containertoepassing voor een SWF-bestand van derden of de zelfstandige Flash Player, kunt u deze nog steeds gebruiken als functie binnen het pakket flash.system.*

De klasse ExternalInterface is een subsysteem waarmee u gemakkelijk met JavaScript op een HTML-pagina kunt communiceren vanuit ActionScript en Flash Player.

De klasse ExternalInterface is alleen beschikbaar onder de volgende omstandigheden:

- In alle ondersteunde versies van Internet Explorer voor Windows (5.0 en hoger).
- In elke browser met ondersteuning voor de NPRuntime-interface. Dit zijn momenteel Firefox 1.0 en hoger, Mozilla 1.7.5 en hoger, Netscape 8.0 en hoger en Safari 1.3 en hoger.
- In een AIR-toepassing wanneer het SWF-bestand is ingesloten op een HTML-pagina die wordt weergegeven door het besturingselement HTMLLoader.

In alle andere situaties (wanneer deze bijvoorbeeld in een zelfstandige speler wordt uitgevoerd) retourneert de eigenschap `ExternalInterface.availablefalse`.

Vanuit ActionScript kunt u een JavaScript-functie aanroepen in de HTML-pagina. De externe API biedt de volgende, verbeterde functionaliteit ten opzichte van `fscommand()`:

- U kunt elke JavaScript-functie gebruiken, niet alleen de functies die kunnen worden gebruikt met de functie `fscommand`.
- U kunt elk aantal argumenten doorgeven met elke naam; u bent niet beperkt tot het doorgeven van een opdracht en een argument met een enkele tekenreeks. Hierdoor is de externe API veel flexibeler dan `fscommand()`.
- U kunt verschillende gegevenstypen (zoals Bolean, Number en String) passeren; u bent niet beperkt tot String-parameters.
- U kunt de waarde van een aanroep ontvangen; deze waarde wordt direct geretourneerd aan ActionScript (als de geretourneerde waarde van uw aanroep).

Belangrijk: *Als de naam die is opgegeven voor de Flash Player-instantie in een HTML-pagina (de tag van het `objectid`-kenmerk) een koppelteken bevat (`-`) of andere tekens die worden gedefinieerd als operatoren in JavaScript (zoals `+`, `*`, `/`, `\`, `.` enz.) ExternalInterface-oproepen vanaf ActionScript werken niet als de containerinternetpagina wordt weergegeven in Internet Explorer. Als de HTML-tags die de Flash Player-instantie definiëren (de `object`- en `embed`-tags) zijn genest in een HTML `form`-tag, werken ExternalInterface-oproepen vanaf ActionScript niet.*

De klasse ExternalInterface gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Communicatie tussen ActionScript en de containertoepassing kan op twee manieren plaatsvinden: ActionScript kan code (bijvoorbeeld een JavaScript-functie) aanroepen die in de container is gedefinieerd, of code in de container kan een ActionScript-functie aanroepen die als aanroepbaar is aangewezen. In beide gevallen kan informatie worden verstuurd naar de aangeroepen code en kunnen de resultaten worden geretourneerd aan de code die de aanroep uitvoert.

Om deze communicatie mogelijk te maken, bevat de klasse ExternalInterface twee statische eigenschappen en twee statische methoden. Deze eigenschappen en methoden worden gebruikt om informatie te verzamelen over de externe interfaceverbinding, om de code uit te voeren in de container vanuit ActionScript en om ActionScript-functies beschikbaar te maken om te worden aangeroepen door de container.

Informatie ophalen over de externe container

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De eigenschap `ExternalInterface.available` geeft aan of Flash Player zich in een container bevindt die een externe interface biedt. Als de externe interface beschikbaar is, is deze eigenschap `true`; anders is deze `false`. Voordat u gebruik maakt van andere willekeurige functies uit de klasse ExternalInterface, moet u altijd als volgt controleren of de huidige container communicatie met de externe interface ondersteunt:

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

Opmerking: *De eigenschap `ExternalInterface.available` meldt of het huidige type container connectiviteit met ExternalInterface ondersteunt. Het geeft niet aan of JavaScript in de huidige browser is ingeschakeld.*

Met de eigenschap `ExternalInterface.objectID` kunt u de unieke id van de Flash Player-instantie bepalen (met name het kenmerk `id` van de tag `object` in Internet Explorer of het kenmerk `name` van de tag `embed` in browsers die gebruik maken van de NPRuntime-interface). Deze unieke id vertegenwoordigt het huidige SWF-document in de browser en kan worden gebruikt als verwijzing naar het SWF-document; bijvoorbeeld wanneer een JavaScript-functie binnen een containerwebpagina wordt aangeroepen. Wanneer de Flash Player-container geen webbrowser is, is deze eigenschap `null`.

Externe code aanroepen vanuit ActionScript

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De methode `ExternalInterface.call()` voert de code uit binnen de containertoepassing. Hiervoor is ten minste één parameter nodig, een tekenreeks met de naam van de functie die vanuit de containertoepassing moet worden aangeroepen. Extra parameters die worden doorgegeven aan de methode `ExternalInterface.call()` worden doorgegeven aan de container als parameters van de functieaanroep.

```
// 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);
```

Wanneer de container een HTML-pagina is, roept deze methode de JavaScript-functie met de opgegeven naam aan, die gedefinieerd moet worden in een element `script` in de HTML-pagina. De geretourneerde waarde van de JavaScript-functie wordt doorgegeven aan ActionScript.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

Als de container een andere ActiveX-container is, zorgt deze methode ervoor dat het ActiveX-besturingselement voor Flash Player de gebeurtenis `FlashCall` verzendt. De opgegeven functienaam en parameters worden door Flash Player met serienummering gecodeerd in een XML-tekenreeks. De container kan deze informatie benaderen in de eigenschap `request` van het gebeurtenisobject en aan de hand hiervan bepalen hoe deze zijn eigen code uitvoert. Om een waarde naar ActionScript te retourneren, roept de containercode de methode `SetReturnValue()` van het ActiveX-object aan, waarbij het resultaat (met serienummering gecodeerd in een XML-tekenreeks) als parameter van deze methode wordt doorgegeven. Zie "XML-indeling van externe API" op pagina 895 voor meer informatie over de XML-indeling die voor deze communicatie wordt gebruikt.

Ongeacht of de container een webbrowser of een andere ActiveX-container is: als de aanroep mislukt of als de containermethode geen geretourneerde waarde opgeeft, wordt `null` geretourneerd. Als de betreffende omgeving behoort tot een beveiligingssandbox waartoe de aanroepcode geen toegang heeft, genereert de methode `ExternalInterface.call()` een uitzondering SecurityError. U kunt dit omzeilen door in de betreffende omgeving een juiste waarde in te stellen voor `allowScriptAccess`. Als u bijvoorbeeld de waarde van `allowScriptAccess` in een HTML-pagina wilt wijzigen, kunt u het juiste kenmerk in de tags `object` en `embed` bewerken.

ActionScript-code vanuit de container aanroepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een container kan alleen ActionScript-code in een functie aanroepen; een container kan verder geen ActionScript-code aanroepen. Als u een ActionScript-functie wilt aanroepen vanuit de containertoepassing, moet u het volgende doen: de functie registreren bij de klasse ExternalInterface en deze vervolgens aanroepen vanuit de code van de container.

Eerst moet u uw ActionScript-functie registreren, om aan te geven dat deze beschikbaar moet zijn voor de container. Voer de methode `ExternalInterface.addCallback()` als volgt uit:

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

Voor de methode `addCallback()` zijn twee parameters nodig. De eerste, een functienaam als tekenreeks, is de naam waarmee de functie wordt herkend door de container. De tweede parameter is de daadwerkelijke ActionScript-functie, die wordt uitgevoerd wanneer de container de gedefinieerde functienaam aanroept. Omdat deze namen verschillen, kunt u een functienaam opgeven die wordt gebruikt door de container, ook als de daadwerkelijke ActionScript-functie een andere naam heeft. Dit is met name zinvol als de functienaam onbekend is; bijvoorbeeld als een anonieme functie is opgegeven, of als de aan te roepen functie bij uitvoering wordt bepaald.

Nadat een ActionScript-functie is geregistreerd met de klasse ExternalInterface, kan de container de functie daadwerkelijk aanroepen. De wijze waarop hangt af van het type container. De ActionScript-functie wordt bijvoorbeeld in een JavaScript-code in een webbrowser aangeroepen met behulp van de geregistreerde functienaam, ook al is dit een methode van het Flash Player-browserobject (dat wil zeggen een methode van het JavaScript-object dat de tag `object` of `embed` vertegenwoordigt). |Met andere woorden worden er parameters doorgegeven en een resultaat geretourneerd, ook al wordt er een lokale functie aangeroepen.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

Wanneer u een ActionScript-functie aanroept in een SWF-bestand dat wordt uitgevoerd in een bureaubladtoepassing, moeten de geregistreerde functienaam en mogelijke parameters in serie in een XML-tekenreeks worden gecodeerd. De aanroep wordt in dat geval daadwerkelijk uitgevoerd door de methode `CallFunction()` van het ActiveX-besturingselement aan te roepen met de XML-tekenreeks als parameter. Zie "XML-indeling van externe API" op pagina 895 voor meer informatie over de XML-indeling die voor deze communicatie wordt gebruikt.

In beide gevallen wordt de geretourneerde waarde van de ActionScript-functie doorgegeven aan de containercode, ofwel rechtstreeks als waarde wanneer de aanroeper een JavaScript-code in een browser is, ofwel gecodeerd met serienummering in een XML-tekenreeks wanneer de aanroeper een ActiveX-container is.

XML-indeling van externe API

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Bij communicatie tussen ActionScript en een toepassing die het ActiveX-besturingselement voor Shockwave Flash host, wordt gebruikgemaakt van een specifieke XML-indeling om functieaanroepen en waarden te coderen. De externe API maakt gebruik van twee onderdelen van de XML-indeling. Een onderdeel vertegenwoordigt functieaanroepen. Een ander onderdeel vertegenwoordigt individuele waarden; deze indeling wordt gebruikt voor parameters in functies en voor geretourneerde waarden voor functies. De XML-indeling voor functieaanroepen wordt gebruikt voor aanroepen naar en vanuit ActionScript. Voor een functieaanroep vanuit ActionScript geeft Flash Player de XML-tekenreeks door aan de container; voor een aanroep vanuit de container verwacht Flash Player dat de containertoepassing een XML-tekenreeks in deze indeling doorgeeft. Het volgende XML-fragment toont een voorbeeld van een functieaanroep in XML-indeling:

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

De hoofdnode is de node `invoke`. Deze heeft twee kenmerken: `name` geeft de naam van de aan te roepen functie weer en `returntype` is altijd `xml`. Als de functieaanroep parameters bevat, heeft de node `invoke` een onderliggende node `arguments`, waarvan de onderliggende nodes de parameterwaarden zijn die zijn opgemaakt volgens de hieronder uitgelegde individuele waardenindeling.

Individuele waarden, waaronder functieparameters en geretourneerde waarden voor functies, maken gebruik van een indelingsprotocol dat behalve de daadwerkelijke waarden ook informatie over het gegevenstype bevat. De volgende tabel bevat ActionScript-klassen en de XML-indeling, die gebruikt wordt om waarden van dit gegevenstype te coderen:

ActionScript-klasse/waarde	**C#-klasse/waarde**	**Indeling**	**Opmerkingen**
null	null	<null/>	
Boolean `true`	bool true	<true/>	
Boolean `false`	bool false	<false/>	
String	string	<string>tekenreekswaarde</string>	
Number, int, uint	single, double, int, uint	`<number>27.5</number>` `<number>-12</number>`	

Array (elementen kunnen gemengde typen zijn)	Een verzameling waarbij elementen van gemengde typen zijn toegestaan, zoals ArrayList of object[]	`<array>`
		`<property id="0">`
		`<number>27.5</number>`
		`</property>`
		`<property id="1">`
		`<string>Hello there!</string>`
		`</property>`
		`...`
		`</array>`
			De node `property` definieert individuele elementen en het kenmerk `id` is de numerieke, op nul gebaseerde index.

Object	Een dictionary met tekenreekssleutels en objectwaarden, zoals een HashTable met tekenreekssleutels	`<object>`
		`<property id="name">`
		`<string>John Doe</string>`
		`</property>`
		`<property id="age">`
		`<string>33</string>`
		`</property>`
		`...`
		`</object>`
			De node `property` definieert individuele eigenschappen en het kenmerk `id` is de eigenschapnaam (een tekenreeks).

Andere ingebouwde of aangepaste klassen		`<null/> or` `<object></object>`	ActionScript codeert andere objecten als null of als een leeg object. In beide gevallen gaan eigenschapwaarden verloren.

Opmerking: *Deze tabel toont als voorbeeld naast ActionScript-klassen equivalente C#-klassen; de externe API kan echter worden gebruikt om te communiceren met een willekeurige programmeertaal of uitvoering die ActiveX-besturingselementen ondersteunt en is niet beperkt tot C#-toepassingen.*

Wanneer u met een containertoepassing van ActiveX met de externe API zelf toepassingen maakt, vindt u het wellicht handig om een proxy te schrijven die native functieaanroepen omzet in een XML-indeling met serienummering. Zie In de klasse ExternalInterfaceProxy voor een voorbeeld van een proxyklasse in C#.

Voorbeeld van externe API: communicatie tussen ActionScript en JavaScript in een webbrowser

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Deze voorbeeldtoepassing toont geschikte technieken die communicatie tussen ActionScript en JavaScript in een webbrowser mogelijk maken, binnen de context van een Instant Messaging-toepassing waarmee een gebruiker kan chatten met zichzelf (vandaar de naam van de toepassing: Introvert IM). Met behulp van de externe API worden berichten verzonden tussen een HTML-formulier in de webpagina en een SWF-interface. De technieken die in dit voorbeeld worden getoond, zijn onder meer:

- communicatie op correcte wijze starten, door te controleren of de browser klaar is voor communicatie alvorens communicatie in te stellen.
- Controleren of de container de externe API ondersteunt.
- JavaScript-functies aanroepen vanuit ActionScript, parameters doorgeven en als reactie waarden ontvangen.
- ActionScript-methoden beschikbaar maken om te worden aangeroepen door JavaScript en deze aanroepen uitvoeren.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De bestanden voor de Introvert IM-toepassing vindt u in de map Samples/IntrovertIM_HTML. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
IntrovertIMApp.fla of IntrovertIMApp.mxml	Het hoofdtoepassingsbestand voor Flash (FLA) of Flex (MXML).
com/example/programmingas3/introvertIM/IMManager.as	De klasse die communicatie tussen ActionScript en de container tot stand brengt en deze beheert.
com/example/programmingas3/introvertIM/IMMessageEvent.as	Type aangepaste gebeurtenis, verzonden door de klasse IMManager wanneer een bericht wordt ontvangen door de container.
com/example/programmingas3/introvertIM/IMStatus.as	Opsomming waarvan de waarden de verschillende waarden vertegenwoordigen van de status 'beschikbaarheid' die binnen deze toepassing kunnen worden geselecteerd.
html-flash/IntrovertIMApp.html of html-template/index.template.html	De HTML-pagina voor de Flash-toepassing (html-flash/IntrovertIMApp.html) of de sjabloon die wordt gebruikt om de HTML-containerpagina voor de toepassing voor Adobe Flex (html-template/index.template.html) te maken. Dit bestand bevat alle JavaScript-functies die het containergedeelte van de toepassing vormen.

Voorbereiden op communicatie via de ActionScript-browser

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een van de meest gebruikte toepassingen van de externe API is om ActionScript-toepassingen te laten communiceren met een webbrowser. ActionScript-methoden kunnen met de externe API codes aanroepen die zijn geschreven in JavaScript en omgekeerd. Gezien de complexiteit van browsers en de wijze waarop deze pagina's intern renderen, kan niet worden gegarandeerd dat een SWF-document zijn callbacks registreert voordat het eerste JavaScript op de HTML-pagina wordt uitgevoerd. Om deze reden moet uw SWF-document, voordat hierin functies vanuit JavaScript worden aangeroepen, altijd eerst de HTML-pagina aanroepen en melden dat het SWF-document gereed is om verbindingen te accepteren.

De Introvert IM bepaalt bijvoorbeeld aan de hand van een reeks stappen die worden uitgevoerd door de klasse IMManager, of de browser gereed is voor communicatie en bereidt het SWF-bestand voor op communicatie. De eerste stap, waarin wordt bepaald of de browser gereed is voor communicatie, vindt als volgt plaats in de constructor IMManager:

```
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.");
    }
}
```

Ten eerste controleert de code met behulp van de eigenschap `ExternalInterface.available` of de externe API beschikbaar is in de huidige container. Als dit het geval is, start de code het proces waarbij communicatie wordt ingesteld. Omdat er zich uitzonderingen en andere fouten kunnen voordoen wanneer u communicatie tot stand probeert te brengen met een externe toepassing, is de code opgenomen in een blok `try` (voor een beknopte lijst zijn de overeenkomstige blokken `catch` weggelaten).

De volgende code roept de methode `isContainerReady()` aan:

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

De methode `isContainerReady()` maakt op zijn beurt gebruik van de methode `ExternalInterface.call()` om de JavaScript-functie `isReady()` als volgt aan te roepen:

```
<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>
```

De functie `isReady()` retourneert alleen de waarde van de variabele `jsReady`. Deze variabele is aanvankelijk `false`; nadat de gebeurtenis `onload` van de webpagina geactiveerd is, wordt de waarde van de variabele gewijzigd in `true`. Met andere woorden: als ActionScript de functie `isReady()` aanroept voordat de pagina is geladen, retourneert JavaScript `false` naar `ExternalInterface.call("isReady")`; als gevolg hiervan retourneert de methode ActionScript `isContainerReady()false`. Nadat de pagina is geladen, retourneert de JavaScript-functie `isReady()true`, zodat de ActionScript-methode `isContainerReady()` eveneens `true` retourneert.

In de constructor IMManager gebeurt vervolgens één van de volgende twee dingen, afhankelijk van de gereedheid van de container. Als `isContainerReady()true` retourneert, roept de code alleen de methode `setupCallbacks()` aan, die het proces waarbij communicatie met JavaScript wordt ingesteld, afrondt. Als daarentegen `isContainerReady()false` retourneert, wordt het proces gepauzeerd. Er wordt een object Timer gemaakt dat de methode `timerHandler()` iedere 100 milliseconden als volgt aanroept:

```
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();
    }
}
```

Steeds als de methode `timerHandler()` wordt aangeroepen, controleert deze opnieuw het resultaat van de methode `isContainerReady()`. Nadat de container is geïnitialiseerd, retourneert deze methode `true`. De code stopt daarna de Timer en roept de methode `setupCallbacks()` aan om het proces te beëindigen waarbij communicatie met de browser wordt ingesteld.

ActionScript-methoden beschikbaar maken voor JavaScript

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zoals uit het vorige voorbeeld blijkt, wordt de methode `setupCallbacks()` aangeroepen, nadat de code heeft vastgesteld dat de browser gereed is. Deze methode bereidt ActionScript voor om aanroepen te ontvangen vanuit JavaScript, zoals hier wordt getoond:

```
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");
}
```

De methode `setCallBacks()` voltooit de taak waarbij communicatie met de container wordt voorbereid, door `ExternalInterface.addCallback()` aan te roepen en de twee methoden te registreren die beschikbaar zijn om vanuit JavaScript te worden aangeroepen. In deze code is de eerste parameter (de naam waaronder de methode bekend is bij JavaScript, `"newMessage"` en `"getStatus"`) gelijk aan de naam van de methode in ActionScript. (In dit geval leverde het geen voordeel op om verschillende namen te gebruiken, zodat voor het gemak dezelfde naam opnieuw werd gebruikt.) Tot slot wordt de methode `ExternalInterface.call()` gebruikt om de JavaScript-functie `setSWFIsReady()` aan te roepen, die aan de container meldt dat de ActionScript-functies zijn geregistreerd.

Communicatie vanuit ActionScript naar de browser

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De toepassing Introvert IM toont een reeks voorbeelden waarbij JavaScript-functies in de containerpagina worden aangeroepen. In het meest eenvoudige geval (een voorbeeld van de methode `setupCallbacks()`) wordt de functie JavaScript `setSWFIsReady()` aangeroepen zonder dat er parameters worden doorgegeven of er als reactie een waarde wordt ontvangen:

```
ExternalInterface.call("setSWFIsReady");
```

In een ander voorbeeld van de methode `isContainerReady()` roept ActionScript de functie `isReady()` aan en ontvangt als reactie een Booleaanse waarde:

```
var result:Boolean = ExternalInterface.call("isReady");
```

U kunt ook parameters doorgeven aan JavaScript-functies die gebruik maken van de externe API. Zie bijvoorbeeld de methode `sendMessage()` van de klasse IMManager, die wordt aangeroepen wanneer de gebruiker een nieuw bericht verstuurt naar zijn of haar 'conversatiepartner':

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

Ook hier wordt `ExternalInterface.call()` weer gebruikt om de opgegeven JavaScript-functie aan te roepen, waarbij het nieuwe bericht aan de browser wordt doorgegeven. De melding zelf wordt doorgegeven als aanvullende parameter aan `ExternalInterface.call()` en wordt als gevolg hiervan doorgegeven als parameter aan de JavaScript-functie `newMessage()`.

ActionScript-code vanuit JavaScript aanroepen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Communicatie werkt altijd in twee richtingen en de toepassing Introvert IM vormt hierop geen uitzondering. De IM client van Flash Player roept niet alleen JavaScript aan voor het verzenden van berichten, maar het HTML-formulier roept de JavaScript-code aan om zowel berichten te verzenden als om informatie van het SWF-bestand op te vragen. Wanneer het SWF-bestand bijvoorbeeld aan de container meldt dat het contact tot stand heeft gebracht en gereed is om te communiceren, roept de browser eerst de methode `getStatus()` van de klasse IMManager aan, om de beschikbaarheidsstatus van de gebruiker op te halen uit de SWF IM-client. Dit vindt als volgt plaats op de webpagina in de functie `updateStatus()`:

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

De code controleert de waarde van de variabele `swfReady`, die bijhoudt of het SWF-bestand aan de browser heeft doorgegeven dat de methoden ervan zijn geregistreerd met de klasse ExternalInterface. Als het SWF-bestand gereed is communicatie te ontvangen, roept de volgende regel (`var currentStatus =` ...) de methode `getStatus()` daadwerkelijk in de klasse IMManager aan. In deze coderegel gebeuren drie dingen:

- De JavaScript-functie `getSWF()` wordt aangeroepen, die een verwijzing naar het JavaScript-object retourneert die het SWF-bestand vertegenwoordigt. De parameter die is doorgegeven aan `getSWF()` bepaalt welk browserobject wordt geretourneerd wanneer er zich meer dan één SWF-bestand in een HTML-pagina bevindt. De waarde die wordt doorgegeven aan deze parameter moet overeenkomen met het kenmerk `id` van de tag `object` en het kenmerk `name` van de tag `embed` die wordt gebruikt om het SWF-bestand in te sluiten.

- De methode `getStatus()` wordt met de verwijzing naar het SWF-bestand aangeroepen, hoewel dit een methode van het SWF-object is. In dit geval wordt gebruik gemaakt van de functienaam "`getStatus`", omdat dit de naam is waarmee de ActionScript-functie met `ExternalInterface.addCallback()` is geregistreerd.
- De ActionScript-methode `getStatus()` retourneert een waarde en deze waarde wordt toegewezen aan de variabele `currentStatus`, die daarop wordt toegewezen aan de inhoud (de eigenschap `value`) van het tekstveld `status`.

Opmerking: *Als u de code volgt, is u waarschijnlijk al opgevallen dat in de broncode voor de `updateStatus()`-functie, de coderegel die de `getSWF()`-functie aanroept, als volgt is geschreven: var currentStatus = getSWF("${application}").getStatus(De tekst `${application}` is een plaatsaanduiding in het HTML-paginasjabloon; wanneer Adobe Flash Builder de werkelijke HTML-pagina voor de toepassing genereert, wordt deze plaatsaanduidingstekst vervangen door dezelfde tekst die wordt gebruikt als het `id`-kenmerk van het `object`-tag en het `name`-kenmerk van de `embed`-tag (`IntrovertIMApp` in het voorbeeld). Dat is de waarde die wordt verwacht door de functie `getSWF()`.*

De JavaScript-functie `sendMessage()` laat zien hoe een parameter wordt doorgegeven aan een ActionScript-functie. (`sendMessage()` is de functie die wordt aangeroepen wanneer de gebruiker op de knop Verzenden drukt op de HTML-pagina.)

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

De ActionScript-methode `newMessage()` verwacht één parameter, zodat de JavaScript-variabele `message` wordt doorgegeven aan ActionScript door deze te gebruiken als een parameter in de methodeaanroep `newMessage()` in de JavaScript-code.

Type browser detecteren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De wijze waarop browsers inhoud benaderen, kan verschillen. Het is daarom belangrijk dat JavaScript altijd wordt gebruikt om te detecteren welke browser de gebruiker uitvoert en om de film met behulp van het venster- of documentobject te benaderen volgens de browserspecifieke syntaxis, zoals wordt weergegeven in de JavaScript-functie `getSWF()` in dit voorbeeld:

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

Als uw script het type browser van de gebruiker niet detecteert, vertonen de SWF-bestanden tijdens het afspelen in een HTML-container mogelijk een ander gedrag dan de gebruiker verwacht.

Hoofdstuk 48: XML-handtekeningvalidatie in AIR

Adobe AIR 1.5 of hoger

Gebruik de klassen in de Adobe® AIR® XMLSignatureValidator API om digitale handtekeningen te valideren in overeenstemming met een subset met de W3C-aanbeveling voor XML-Signature Syntax and Processing (http://www.w3.org/TR/xmldsig-core/). XML-handtekeningen kunnen worden gebruikt om de integriteit en de identiteit van de ondertekenaar van gegevens of informatie te verifiëren.

Met XML-handtekeningen kunnen berichten of bronnen worden gevalideerd die door uw toepassing worden gedownload. Als uw toepassing bijvoorbeeld diensten levert op abonnementbasis, kunt u de voorwaarden van het abonnement opnemen in een getekend XML-document. Als iemand het document met abonnementvoorwaarden gewijzigd had, zou de validatie mislukken.

U zou een XML-handtekening kunnen gebruiken om gedownloade bronnen te helpen valideren die door uw toepassing worden gebruikt, door een getekend manifest op te nemen dat samenvattingen van deze bronnen bevat. De toepassing kan dan controleren of de bronnen niet zijn gewijzigd, door de samenvatting in het getekende bestand te vergelijken met de samenvatting die vanuit de geladen bytes is berekend. Dit is met name waardevol wanneer de gedownloade bron een SWF-bestand of andere actieve inhoud is die u in de beveiligingssandbox van de toepassing wilt uitvoeren.

Basisbeginselen van de validatie van XML-handtekeningen

Adobe AIR 1.5 of hoger

Lees voor een snelle uitleg en codevoorbeelden van het valideren van XML-handtekeningen, de volgende snelstartartikelen in de Adobe Developer Connection:

- XML-handtekeningen maken en valideren (Flex)
- Creating and validating XML signatures (Flash)

Adobe® AIR® biedt de XMLSignatureValidator-klasse en de IURIDereferencer-interface voor het valideren van XML-handtekeningen. De XML-syntaxis die wordt geaccepteerd door de klasse XMLSignatureValidator is een subset van de W3C-aanbevelingen voor de syntaxis en verwerking van XML-handtekeningen. (Omdat alleen maar een subset van de aanbevelingen wordt ondersteund, kunnen niet alle geldige handtekeningen worden gevalideerd.) AIR is niet voorzien van een API voor het maken van XML-handtekeningen.

Klassen voor XML-handtekeningvalidatie

Adobe AIR 1.5 of hoger

De API voor XML-handtekeningvalidatie is voorzien van de volgende klassen:

Pakket	Klassen
flash.security	• XMLSignatureValidator • IURIDereferencer (interface) Tekenreeksconstanten voor XMLSignatureValidator worden gedefinieerd in de volgende klassen: • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Gebeurtenis • ErrorEvent

De klassen voor XML-handtekeningvalidatie gebruiken

Adobe AIR 1.5 of hoger

Om de klasse XMLSignatureValidator te gebruiken om een XML-handtekening te valideren, moet u:

- Een object XMLSignatureValidator maken
- Een implementatie van de IURIDereferencer-interface bieden. Het object XMLSignatureValidator roept de methode `dereference()` van IURIDereferencer aan en geeft de URI voor iedere referentie in een handtekening door. De methode `dereference()` moet de URI oplossen en de gegevens waarnaar wordt verwezen, retourneren (deze kunnen zich bevinden in hetzelfde document als de handtekening of in een externe bron).
- Geef de vertrouwensinstellingen voor certificaten, de instellingen voor de intrekkingscontrole van certificaten en de instellingen voor de validatie van referenties voor het XMLSignatureValidator-object op zoals u deze wilt gebruiken voor uw toepassing.
- Voeg gebeurtenislisteners voor de gebeurtenissen `complete` en `error` toe.
- Roep de methode `verify()` aan, en geef de te valideren handtekening door.
- Handel de gebeurtenissen `complete` en `error` af en interpreteer de resultaten.

In het volgende voorbeeld wordt een functie `validate()` geïmplementeerd die de geldigheid van een XML-handtekening verifieert. De eigenschappen van XMLSignatureValidator zijn zodanig ingesteld dat het ondertekenende certificaat zich moet bevinden in de vertrouwde opslag van het systeem, of zijn gekoppeld aan een certificaat in de vertrouwde opslag. Er wordt verder in dit voorbeeld van uitgegaan dat een toepasselijke IURIDereferencer-klasse met de naam *XMLDereferencer* bestaat.

```
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);
}
```

Validatieproces van XML-handtekeningen

Adobe AIR 1.5 of hoger

Wanneer u de XMLSignatureValidator-methode `verify()` aanroept, voert AIR de volgende stappen uit:

- De runtime controleert de kryptografische integriteit van de handtekening met behulp van de openbare sleutel van het ondertekenende certificaat.
- De runtime bevestigt de kryptografische integriteit, identiteit en betrouwbaarheid van het certificaat op basis van de huidige instellingen van het XMLSignatureValidator-object.

  Het vertrouwen dat wordt gegeven aan het ondertekenende certificaat is de sleutel voor de integriteit van het validatieproces. De handtekeningvalidatie wordt uitgevoerd aan de hand van een goedgedefinieerd cryptografisch proces, maar de betrouwbaarheid van het ondertekenende certificaat kan niet worden beoordeeld aan de hand van een algoritme.

  In het algemeen kunt u op drie manieren bepalen of een certificaat betrouwbaar is:

  - Door te vertrouwen op certificeringsinstanties en het vertrouwensarchief van het besturingssysteem.

- U kunt bij de ondertekenaar een kopie van het certificaat opvragen, een ander certificaat dat fungeert als vertrouwensanker voor het certificaat, of voldoende informatie om het certificaat op betrouwbare wijze te kunnen identificeren, bijvoorbeeld de openbare sleutel.
- U kunt de eindgebruikers van uw toepassing vragen of zij het certificaat vertrouwen. Zo'n vraag is ongeldig bij zelfondertekende certificaten, aangezien de identificerende informatie in dergelijke certificaten van nature onbetrouwbaar is.

- De runtime controleert de kryptografische integriteit van de ondertekende gegevens.

  De ondertekende gegevens worden geverifieerd met behulp van uw IURIDereferencer-implementatie. Voor elke referentie in het handtekeningdocument wordt de methode `dereference()` van de IURIDereferencer-implementatie aangeroepen. De gegevens die de methode `dereference()` retourneert, worden gebruikt om de referentiesamenvatting te berekenen. De samenvattingswaarde wordt vergeleken met de samenvatting die in het handtekeningdocument is opgenomen. Als deze twee samenvattingen overeenstemmen, zijn de gegevens niet gewijzigd sinds ze werden ondertekend.

  Als u vertrouwt op de validatieresultaten van een XML-handtekening, moet u er wel rekening mee houden dat alleen datgene wat is ondertekend, ook veilig is. Neem bijvoorbeeld een ondertekend manifest waarin de bestanden in een pakket worden aangegeven. Wanneer de XMLSignatureValidator de handtekening verifieert, wordt alleen gecontroleerd of het manifest zelf ongewijzigd is. De gegevens in de bestanden zijn niet ondertekend; de handtekening zelf zal dus nog steeds worden gevalideerd, ook wanneer bestanden waarnaar wordt verwezen in het manifest, gewijzigd of verwijderd zijn.

  Opmerking: *Om bestanden in zo'n manifest te verifiëren, kunt u de samenvatting van de bestandsgegevens berekenen (met gebruikmaking van hetzelfde hash-algoritme dat in het manifest is gebruikt) en het resultaat vergelijken met de samenvatting die is opgeslagen in het ondertekende manifest. In sommige gevallen moet u ook controleren op de aanwezigheid van extra bestanden.*

Validatieresultaten interpreteren

Adobe AIR 1.5 of hoger

De validatieresultaten worden gerapporteerd door de statuseigenschappen van het XMLSignatureValidator-object. Deze eigenschappen kunnen worden gelezen nadat het validator-object de gebeurtenis *complete* verzendt. De status kan een van de volgende vier waarden hebben: `validityStatus`, `digestStatus`, `identityStatus` en `referencesStatus`.

De eigenschap validityStatus

Adobe AIR 1.5 of hoger

De eigenschap `validityStatus` rapporteert de algehele geldigheid van de handtekening. De `validityStatus` is afhankelijk van de toestand van de drie andere statuseigenschappen en kan een van de volgende waarden hebben:

- `valid` — Als `digestStatus`, `identityStatus` en `referencesStatus` allemaal `geldig` zijn.
- `invalid` — Als een of meer van de individuele statuseigenschappen `ongeldig` zijn.
- `unknown` — Als een of meer afzonderlijke statuseigenschappen `unknown` is en geen enkele status `invalid` is.

De eigenschap digestStatus

Adobe AIR 1.5 of hoger

De eigenschap `digestStatus` rapporteert de resultaten van de cryptografische verificatie van de berichtsamenvatting. De eigenschap `digestStatus` kan een van de volgende waarden hebben:

- `valid` — Als het handtekeningdocument zelf sinds de ondertekening niet meer is gewijzigd.
- `invalid` — Als het handtekeningdocument is gewijzigd of beschadigd.
- `unknown` — Als de methode `verify()` niet is voltooid zonder fout.

De eigenschap identityStatus

Adobe AIR 1.5 of hoger

De eigenschap `identityStatus` rapporteert de status van het ondertekenende certificaat. De waarde van deze eigenschap is afhankelijk van verschillende factoren, waaronder:

- de cryptografische integriteit van het certificaat
- is het certificaat verlopen dan wel ingetrokken
- wordt het certificaat vertrouwd op de huidige computer
- de status van het XMLSignatureValidator-object (bijvoorbeeld of er extra certificaten zijn toegevoegd voor het opbouwen van de vertrouwensketen, of die certificaten worden vertrouwd, en de waarden van de eigenschappen `useSystemTrustStore` en `revocationCheckSettings`)

De eigenschap `identityStatus` kan de volgende waarden hebben:

- `valid` — Het ondertekenende certificaat kan alleen als geldig worden beschouwd als het voldoet aan de volgende voorwaarden:
  - Het ondertekenende certificaat moet ongewijzigd zijn.
  - Het ondertekenende certificaat mag niet zijn verlopen of ingetrokken, behalve wanneer de handtekening een geldige tijdstempel bevat. Als de handtekening een tijdstempel bevat, wordt het certificaat als geldig beschouwd als het geldig was op het moment dat het document werd ondertekend. (Het certificaat dat door de tijdstempelservice is gebruikt om het tijdstempel te plaatsen, moet zijn gekoppeld aan een vertrouwd basiscertificaat op de computer van de gebruiker.)
  - Het ondertekenende certificaat wordt vertrouwd. Het certificaat wordt vertrouwd als het certificaat zich bevindt in de vertrouwde opslag van het systeem of als het certificaat is gekoppeld aan een ander certificaat in de vertrouwde opslag van het systeem en u de eigenschappen `useSystemTrustStore` instelt op waar. U kunt ook een certificaat instellen als vertrouwd met de hulp van de methode `addCertificate()` van het XMLSignatureValidator-object.
  - Het certificaat moet werkelijk het ondertekenende certificaat zijn.
- `invalid` — Het certificaat is verlopen of ingetrokken (en er is geen tijdstempel aanwezig die bewijst dat het certificaat geldig was toen het werd ondertekend), of het certificaat is gewijzigd.
- `unknown` — Als het certificaat niet ongeldig is, maar ook niet wordt vertrouwd. Zelfondertekende certificaten worden bijvoorbeeld gerapporteerd als `unknown` (behalve als ze expliciet worden vertrouwd). De `identityStatus` wordt ook gerapporteerd als `unknown` als de methode `verify()` niet zonder fout is voltooid of als de identiteit niet is gecontroleerd omdat de samenvatting van de handtekening ongeldig is.

De eigenschap referencesStatus

Adobe AIR 1.5 of hoger

De eigenschap `referencesStatus` rapporteert de cryptografische integriteit van de referenties in het SignedData-element van de handtekening.

- `valid` — Als de berekende samenvatting van iedere referentie in de handtekening overeenkomt met de corresponderende samenvatting die is geregistreerd in de XML-handtekening. De status `valid` geeft aan dat de ondertekende gegevens niet zijn gewijzigd.
- `invalid` — Als een berekende samenvatting niet overeenstemt met de corresponderende samenvatting in de handtekening.
- `unknown` — Als de referentiesamenvattingen niet zijn gecontroleerd. De referenties worden niet gecontroleerd als de algehele handtekeningsamenvatting de waarde`invalid` heeft of als het ondertekenende certificaat ongeldig is. Als de `identityStatus` de waarde `unknown` heeft, worden de referenties alleen gecontroleerd wanneer `referencesValidationSetting` de waarde `validOrUnknown` heeft.

Informatie over XML-handtekeningen

Adobe AIR 1.5 of hoger

Een XML-handtekening is een digitale handtekening in de XML-syntaxis. De gegevens in een XML-handtekening kunnen worden gebruikt om te valideren dat de ondertekende informatie sinds de ondertekening niet is gewijzigd. Verder geldt dat wanneer een ondertekenend certificaat is uitgegeven door een vertrouwde certificeringsinstantie, de identiteit van de ondertekenaar kan worden geverifieerd via de infrastructuur voor openbare sleutels.

Een XML-handtekening kan worden toegepast op elk soort digitale gegevens (in binaire of XML-indeling). XML-handtekeningen worden in het algemeen gebruikt voor doeleinden als de volgende:

- controleren of externe dan wel gedownloade bronnen zijn gewijzigd
- verifiëren of berichten afkomstig zijn van een bekende bron
- valideren van licenties voor toepassingen of machtigingen in verband met abonnementen

Ondersteunde syntaxis voor XML-handtekeningen

Adobe AIR 1.5 of hoger

AIR ondersteunt de volgende elementen uit de W3C-aanbevelingen voor de syntaxis en verwerking van XML-handtekeningen:

- Alle elementaire syntaxiselementen voor handtekeningen (sectie 4 van de W3C-aanbevelingen); alleen het KeyInfo-element wordt niet volledig ondersteund
- Het KeyInfo-element mag alleen een X509Data-element bevatten
- Een X509Data-element mag alleen een X509Certificate-element bevatten
- De SHA256-samenvattingsmethode
- Het RSA-SHA1 (PKCS1)-ondertekeningsalgoritme
- De "Canonical XML zonder commentaar" methode Canonicalization en algoritme Transform

- De "enveloped signature transform"
- tijdstempels

Het volgende document illustreert een typische XML-handtekening (de meeste cryptografische gegevens zijn verwijderd om het voorbeeld te vereenvoudigen):

```
<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>
```

De belangrijkste elementen van een handtekening zijn:

- SignedInfo — Bevat verwijzingen naar de ondertekende gegevens en de berekende samenvattingswaarden op het moment van ondertekening. De ondertekende gegevens kunnen zelf worden opgenomen in hetzelfde document als de XML-handtekening, of ze kunnen extern zijn.
- SignatureValue — Bevat een samenvatting van het SignedInfo-element, gecodeerd met de persoonlijke sleutel van de ondertekenaar.
- KeyInfo — Bevat het ondertekenende certificaat, evenals aanvullende certificaten die nodig zijn om de vertrouwensketen tot stand te brengen. Hoewel het KeyInfo-element technisch gesproken optioneel is, kan AIR de handtekening niet valideren als het niet is opgenomen.

Er zijn drie algemene typen XML-handtekeningen:

- Enveloped — de handtekening wordt ingevoegd in de XML-gegevens die worden ondertekend.
- Enveloping — de ondertekende XML-gegevens bevinden zich in een Object-element binnen het Signature-element.
- Detached — de ondertekende gegevens zijn extern ten opzichte van de XML-handtekening. De ondertekende gegevens kunnen zich bijvoorbeeld in een extern bestand bevinden. Ze kunnen zich ook in hetzelfde XML-document als de handtekening bevinden, maar niet als bovenliggend of onderliggend element van het Signature-element.

XML-handtekeningen maken gebruik van URI's om te verwijzen naar de ondertekende gegevens. De ondertekenende en validerende toepassingen moeten gebruikmaken van dezelfde conventies voor het oplossen van deze URI's. Wanneer u de klasse XMLSignatureValidator gebruikt, moet u een implementatie van de IURIDereferencer-interface verstrekken. Deze implementatie moet ervoor zorgen dat de URI wordt opgelost en de ondertekende gegevens als ByteArray-object worden geretourneerd. Het geretourneerde ByteArray-object wordt samengevat met behulp van hetzelfde algoritme dat de samenvatting in de handtekening heeft geproduceerd.

Certificaten en vertrouwen

Adobe AIR 1.5 of hoger

Een certificaat bestaat uit een openbare sleutel, informatie die het certificaat identificeert en mogelijk een of meer certificaten die horen bij de uitgevende certificeringsinstantie.

Vertrouwen in een certificaat kan op twee manieren worden bepaald. U kunt vertrouwen tot stand brengen door een kopie van het certificaat op te vragen bij de ondertekenaar, bijvoorbeeld op fysieke media, of via een veilige digitale transmissie, bijvoorbeeld een SSL-transactie. U kunt ook afgaan op een certificeringsinstantie om te bepalen of het ondertekenende certificaat betrouwbaar is.

Als u wilt afgaan op een certificeringsinstantie, moet het ondertekenende certificaat worden uitgegeven door een instantie die wordt vertrouwd op de computer waarop de handtekening wordt gevalideerd. De meeste fabrikanten van besturingssystemen plaatsen de basiscertificaten van een aantal certificeringsinstanties in het vertrouwensarchief van het besturingssysteem. Gebruikers kunnen ook certificaten toevoegen aan en verwijderen uit deze opslag.

Zelfs als een certificaat is uitgegeven door een vertrouwde certificeringsinstantie, moet u nog beslissen of het certificaat hoort bij iemand die u vertrouwt. Vaak wordt deze beslissing doorgegeven naar de eindgebruiker. Wanneer bijvoorbeeld een AIR-toepassing wordt geïnstalleerd, geeft het AIR-installatieprogramma de identificerende informatie van het certificaat van de uitgever weer als de gebruiker wordt gevraagd of deze de toepassing wil installeren. In andere gevallen moet u mogelijk de openbare sleutel of andere certificaatinformatie vergelijken met een lijst met acceptabele sleutels. (Deze lijst moet worden beveiligd, mogelijk door middel van een eigen handtekening of door de lijst op te slaan in de door AIR gecodeerde lokale opslag, zodat er niet met de lijst kan worden geknoeid.)

Opmerking: *U kunt er weliswaar voor kiezen het ondertekenende certificaat te vertrouwen zonder onafhankelijke verificatie, bijvoorbeeld wanneer een handtekening zelf-ondertekend is, maar zo 'n verificatie betekent niet veel wat betreft betrouwbaarheid. Als u niet weet wie een handtekening heeft gemaakt, is de overtuiging dat er niet met de handtekening is geknoeid van weinig of geen waarde. De handtekening kan een correct ondertekende vervalsing zijn.*

Verlopen en intrekking van certificaten

Adobe AIR 1.5 of hoger

Alle certificaten verlopen uiteindelijk. Certificaten kunnen ook worden ingetrokken door de uitgevende certificeringsinstantie, bijvoorbeeld als de persoonlijke sleutel voor het certificaat door een kwaadwillende wordt gewijzigd of gestolen. Als een handtekening is ondertekend met een verlopen of ingetrokken certificaat, wordt de handtekening gerapporteerd als ongeldig, behalve als een tijdstempel is opgenomen als onderdeel van de handtekening. Als er een tijdstempel aanwezig is, valideert de klasse XMLSignatureValidator de handtekening mits het certificaat geldig was op het ogenblik dat het werd ondertekend.

Een tijdstempel is een getekend digitaal bericht van een tijdstempelservice dat bevestigt dat de gegevens op een bepaald tijdstip en een bepaalde datum zijn ondertekend. Tijdstempels worden uitgegeven door tijdstempelinstanties en worden ondertekend door het eigen certificaat van de tijdstempelinstantie. Het tijdstempel kan alleen worden vertrouwd wanneer het certificaat van de tijdstempelinstantie dat is ingesloten in de tijdstempel, wordt vertrouwd op de huidige computer. De XMLSignatureValidator biedt geen API waarmee een ander certificaat kan worden aangewezen voor het valideren van de tijdstempel.

Implementatie van de IURIDereferencer-interface

Adobe AIR 1.5 of hoger

Wanneer u een XML-handtekening wilt valideren, moet u een implementatie van de IURIDereferencer-interface verstrekken. Deze implementatie moet ervoor zorgen dat de URI's binnen de Reference-elementen van een XML-handtekeningdocument worden opgelost en dat de gegevens worden geretourneerd zodat de samenvatting kan worden berekend. De berekende samenvatting wordt vergeleken met de samenvatting in de ondertekening om te bepalen of de gegevens waarnaar wordt verwezen, zijn gewijzigd sinds de handtekening werd gemaakt.

Opmerking: *HTML-gebaseerde AIR-toepassingen moeten een SWF-bibliotheek importeren die een ActionScript-implementatie bevat ten einde XML-handtekeningen te kunnen valideren. De IURIDereferencer-interface kan niet worden geïmplementeerd in JavaScript.*

De IURIDerefencer-interface heeft één methode, `dereference(uri:String)`, die moet worden geïmplementeerd. Het XMLSignatureValidator-object roept deze methode aan voor iedere referentie in de handtekening. De methode moet de gegevens retourneren in een ByteArray-object.

In de meeste gevallen zult u ook eigenschappen of methoden moeten toevoegen waarmee uw dereferencer-object de gegevens waarnaar wordt verwezen, kan vinden. Als de ondertekende gegevens zich bijvoorbeeld bevinden in hetzelfde document als de ondertekening, kunt u een lidvariabele toevoegen die een referentie naar het XML-document biedt. De methode `dereference()` kan deze variabele dan in combinatie met de URI gebruiken om de gegevens waarnaar wordt verwezen, te zoeken. Als de ondertekende gegevens zich bevinden in een map op het lokale bestandssysteem, kan de methode `dereference()` een eigenschap nodig hebben die het pad opgeeft naar de desbetreffende map ten einde de bestanden waarnaar wordt verwezen, op te lossen.

Voor het interpreteren van de URI-tekenreeksen is de XMLSignatureValidator geheel afhankelijk van de dereferencer. De standaardregels voor het oplossen van URI's vindt u in sectie 4.3.3 van de W3C Recommendation for XML Signature Syntax and Processing.

URI's in enveloped handtekeningen oplossen

Adobe AIR 1.5 of hoger

Wanneer een enveloped XML-handtekening wordt gegenereerd, worden de Signature-elementen ingevoegd in de ondertekende gegevens. Als u bijvoorbeeld het volgende bericht ondertekent met behulp van een enveloped handtekening:

```
<message>
    <data>...</data>
</message>
```

Ziet het ontstane ondertekende document er zo uit:

```
<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>
```

U ziet dat de handtekening één enkel Reference-element bevat met een lege tekenreeks als URI. In deze context verwijst een lege tekenreeks naar de hoofdmap van het document.

U ziet ook dat het transformatiealgoritme specificeert dat een enveloped handtekeningtransformatie wordt toegepast. Wanneer een enveloped handtekeningtransformatie is toegepast, verwijdert de XMLSignatureValidator automatisch de handtekening uit het document voordat de samenvatting wordt berekend. Dit betekent dat de dereferencer het Signature-element niet hoeft te verwijderen wanneer de gegevens worden geretourneerd.

In het volgende voorbeeld wordt een dereferencer voor enveloped handtekeningen geïllustreerd:

```
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;
            }
        }
    }
}
```

Deze dereferencer-klasse maakt gebruik van een constructorfunctie met een parameter `signedMessage`, waarmee het enveloped handtekeningdocument beschikbaar wordt gesteld voor de methode `dereference()`. Aangezien de referentie in een enveloped handtekening altijd verwijst naar de hoofdmap van de ondertekende gegevens, schrijft de methode `dereferencer()` het document naar een bytearray, waarna het wordt geretourneerd.

Oplossen van URI's in enveloping en detached handtekeningen

Adobe AIR 1.5 of hoger

Wanneer de ondertekende gegevens zich bevinden in hetzelfde document als de handtekening, gebruiken de URI's in de referenties meestal de XPath- of XPointer-syntaxis om te verwijzen naar de elementen die worden ondertekend. In de W3C-aanbevelingen voor de syntaxis en verwerking van XML-handtekeningen wordt alleen deze syntaxis aanbevolen. Het is dus aan te raden uw implementatie te baseren op de handtekeningen die u naar verwachting aan zult treffen. Zorg ervoor dat u voldoende foutafhandelingsprocedures inbouwt voor de verwerking van syntaxis die niet wordt ondersteund.

De handtekening van een AIR-toepassing is een voorbeeld van een enveloping handtekening. De bestanden in de toepassing worden aangegeven in een Manifest-element. Naar het Manifest-element wordt verwezen in het attribuut Reference URI door middel van de tekenreeks “#PackageContents”, die verwijst naar de ID van het Manifest-element:

```
<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>
```

Een dereferencer voor het valideren van deze handtekening moet de URI-tekenreeks die `"#PackageContents"` uit het Reference-element bevat, accepteren en het Manifest-element in een ByteArray-object retourneren. Het symbool "#" verwijst naar de waarde van een element-ID-attribuut.

In het volgende voorbeeld wordt een dereferencer voor het valideren van handtekeningen voor AIR-toepassingen geïmplementeerd. De implementatie wordt eenvoudig gehouden doordat wordt afgegaan op de bekende structuur van een AIR-handtekening. Een dereferencer voor algemeen gebruik kan aanzienlijk complexer zijn.

```
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;
            }

        }
    }
}
```

Wanneer u dit type handtekening verifieert, worden alleen de gegevens in het Manifest-element gevalideerd. De feitelijke bestanden in het pakket worden in het geheel niet gecontroleerd. Als u wilt controleren of er met de bestanden in het pakket is geknoeid, moet u de bestanden lezen, de SHA256-samenvatting berekenen en het resultaat vergelijken met de samenvatting die is geregistreerd in het manifest. De XMLSignatureValidator controleert dergelijke secundaire referenties niet automatisch.

Opmerking: *Dit voorbeeld wordt alleen gegeven om het validatieproces van handtekeningen te illustreren. Het heeft weinig zin om een AIR-toepassing zijn eigen handtekening te laten valideren. Als er al met de toepassing is geknoeid, kan degene die dit doet de validatiecontrole gewoon verwijderen.*

Samenvattingswaarden voor externe bronnen berekenen

Adobe AIR 1.5 of hoger

AIR is niet voorzien van ingebouwde functies voor het berekenen van SHA256-samenvattingen, maar de Flex SDK is voorzien van een hulpprogrammaklasse SHA256. De SDK omvat ook de Base64 encoder-hulpprogrammaklasse waarmee de berekende samenvatting kan worden vergeleken met de samenvatting die is opgeslagen in de handtekening.

De volgende voorbeeldfunctie leest en valideert de bestanden in een AIR-pakketmanifest:

```
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;
}
```

De functie doorloopt alle referenties in het Manifest-element. Voor iedere referentie wordt de SHA256-samenvatting berekend, gecodeerd in base64-indeling en vergeleken met de samenvatting in het manifest. De URI's in een AIR-pakket verwijzen naar paden die relatief zijn ten opzichte van de toepassingsmap. De paden worden opgelost op basis van de locatie van het handtekeningbestand, dat zich altijd bevindt in de submap META-INF in de toepassingsmap. Houd er rekening mee dat de klasse Flex SHA256 de samenvatting retourneert in de vorm van een reeks hexadecimale tekens. Deze tekenreeks moet worden geconverteerd naar een ByteArray die de bytes bevat die worden vertegenwoordigd door de hexadecimale tekenreeks.

Als u de klassen mx.utils.SHA256 en Base64Encoder in Flash CS4 wilt gebruiken, kunt u deze klassen opzoeken en kopiëren naar de ontwikkelmap van uw toepassing, of kunt u met behulp van de Flex SDK een bibliotheek-SWF compileren die deze klassen bevat.

Oplossen van URI's in detached handtekeningen die verwijzen naar externe gegevens

Adobe AIR 1.5 of hoger

Wanneer een URI verwijst naar een externe bron, moeten de gegevens worden geopend en geladen in een ByteArray-object. Als de URI een absolute URL bevat, hoeft er alleen maar een bestand te worden gelezen of een URL te worden opgevraagd. Wanneer de URI een relatief pad bevat, hetgeen meestal het geval zal zijn, moet uw IURIDereferencer-implementatie een manier bevatten om de paden naar de ondertekende bestanden op te lossen.

In het volgende voorbeeld wordt een File-object gebruikt dat wordt geïnitialiseerd wanneer de dereferencer-instantie wordt geconstrueerd als basis voor het oplossen van ondertekende bestanden.

```
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;
            }
        }
    }
}
```

De functie `dereference()` zoekt eigenlijk alleen het bestand dat is geadresseerd door de referentie-URI, laadt de inhoud van het bestand in een byte-array, en retourneert het ByteArry-object.

Opmerking: *Voordat u de externe referenties gaat valideren, moet u bedenken of uw toepassing mogelijk ontvankelijk is voor een 'phone home' of soortgelijk type aanval vanuit een met verkeerde bedoelingen opgesteld handtekeningdocument.*

Hoofdstuk 49: Clientsysteemomgeving

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In dit hoofdstuk wordt uitgelegd hoe u communiceert met het systeem van de gebruiker. Hier wordt uitgelegd hoe u kunt bepalen welke functies worden ondersteund en hoe u meertalige toepassingen kunt bouwen met behulp van de door de gebruiker geïnstalleerde IME (invoermethode-editor), indien beschikbaar. In dit hoofdstuk worden bovendien voorbeelden gegeven van hoe toepassingsdomeinen doorgaans worden gebruikt.

Meer Help-onderwerpen

flash.system.System

flash.system.Capabilities

Basisbeginselen van de clientsysteemomgeving

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u werkt met meer geavanceerde toepassingen, hebt u soms gedetailleerde informatie nodig over het besturingssysteem van uw gebruikers. Ook dient u soms toegang te hebben tot bepaalde functies van het besturingssysteem. Het flash.system-pakket bevat een verzameling klassen waarmee u toegang krijgt tot functionaliteit op systeemniveau, zoals de volgende:

- Bepalen in welke toepassing en in welk beveiligingsdomein er code wordt uitgevoerd.
- De eigenschappen en mogelijkheden van de Flash-runtime (zoals Flash® Player of Adobe® AIR™) van de gebruiker bepalen, zoals de schermgrootte (resolutie) en de beschikbaarheid van bepaalde functionaliteit, zoals MP3-audio.
- Meertalige websites bouwen met IME.
- Communiceren met de container van Flash-runtime (dit kan een HTML-pagina of een containertoepassing zijn).
- Informatie opslaan op het klembord van de gebruiker.

Het flash.system-pakket bevat ook de klassen IMEConversionMode en SecurityPanel. Deze klassen bevatten statische constanten die u kunt gebruiken in combinatie met respectievelijk de klassen IME en Security.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen die in dit hoofdstuk worden gebruikt:

Besturingssysteem Het hoofdprogramma dat op een computer wordt uitgevoerd en waarbinnen alle andere toepassingen worden uitgevoerd, bijvoorbeeld Microsoft Windows, Mac OS X of Linux®.

clipboard Een container van het besturingssysteem waarin teksten of items die zijn gekopieerd of geknipt worden opgeslagen of van waaruit deze items worden geplakt in toepassingen.

Toepassingsdomein Een mechanisme om de klassen te scheiden die in verschillende SWF-bestanden worden gebruikt, zodat verschillende klassen met dezelfde naam in SWF-bestanden elkaar niet overschrijven.

IME (Input Method Editor ofwel invoermethode-editor) Een programma (of een gereedschap van het besturingssysteem) dat wordt gebruikt om complexe tekens of symbolen in te voeren via een standaardtoetsenbord.

Clientsysteem In programmeertermen is een client het onderdeel van een toepassing (of een complete toepassing) dat wordt uitgevoerd op de computer van een individuele gebruiker en door één individuele gebruiker wordt gebruikt. Het clientsysteem is het onderliggende besturingssysteem op de computer van de gebruiker.

De klasse System gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse System bevat methoden en eigenschappen waarmee u kunt communiceren met het besturingssysteem van de gebruiker en het huidige geheugengebruik van de runtime kunt opvragen. Met de methoden en eigenschappen van de klasse System kunt u bovendien luisteren naar gebeurtenissen `imeComposition`, de runtime instrueren externe tekstbestanden te laden met behulp van de huidige codepagina van de gebruiker of als Unicode, of de inhoud van het klembord van de gebruiker instellen.

Bij uitvoering gegevens ophalen over het systeem van de gebruiker

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de hoeveelheid geheugen (in bytes) bepalen die de runtime momenteel gebruikt door de eigenschap `System.totalMemory` te controleren. Met deze eigenschap kunt u het geheugengebruik volgen en de toepassingen optimaliseren op basis van de wijzigingen in de gebruikte hoeveelheid geheugen. Als een bepaald visueel effect bijvoorbeeld een grote toename van de hoeveelheid geheugen veroorzaakt, kunt u overwegen het effect te wijzigen of te verwijderen.

De eigenschap `System.ime` verwijst naar de op dit moment geïnstalleerde IME (invoermethode-editor). Deze eigenschap stelt u in staat te luisteren naar gebeurtenissen `imeComposition` (`flash.events.IMEEvent.IME_COMPOSITION`) met de methode `addEventListener()`.

De derde eigenschap in de klasse System is `useCodePage`. Wanneer de eigenschap `useCodePage` is ingesteld op `true`, gebruikt de runtime de traditionele codepagina van het besturingssysteem om externe tekstbestanden te laden. Wanneer de eigenschap is ingesteld op `false`, worden externe tekstbestanden in de runtime geïnterpreteerd als Unicode.

Als u `System.useCodePage` instelt op `true`, moet de traditionele codepagina van het besturingssysteem de tekens bevatten die worden gebruikt in het externe tekstbestand. Als dat niet het geval is, wordt de tekst niet weergegeven. Als u bijvoorbeeld een extern tekstbestand laadt dat Chinese tekens bevat, kunnen die tekens niet worden weergegeven op een systeem dat de Nederlandse Windows-codepagina gebruikt omdat die codepagina geen Chinese tekens bevat.

Codeer alle externe tekstbestanden als Unicode en stel `System.useCodePage` standaard in op `false` om ervoor te zorgen dat gebruikers op alle platforms de externe tekstbestanden kunnen weergeven die in uw toepassing worden gebruikt. Op deze manier wordt de tekst in de runtime geïnterpreteerd als Unicode.

Tekst opslaan naar het klembord

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De klasse System bevat een methode met de naam `setClipboard()`, waarmee Flash-runtime de inhoud van het klembord van de gebruiker kan instellen met bepaalde tekenreeks. Vanwege beveiligingsredenen is er geen methode `Security.getClipboard()`. Een dergelijke methode zou kwaadaardige websites mogelijk toegang kunnen geven tot de informatie die het laatst naar het klembord van de gebruiker is gekopieerd.

De volgende code toont hoe een foutbericht naar het klembord van de gebruiker kan worden gekopieerd wanneer een beveiligingsfout optreedt. Een dergelijk foutbericht kan nuttig zijn als de gebruiker een mogelijke fout in een toepassing wil rapporteren.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 en AIR 1.0
U kunt de klasse Clipboard gebruiken voor het lezen en schrijven van klembordgegevens als reactie op een gebruikersgebeurtenis. In AIR is er geen gebruikersgebeurtenis vereist voor code die in de toepassingssandbox wordt uitgevoerd om toegang te krijgen tot het klembord.

De klasse Capabilities gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de klasse Capabilities kunnen ontwikkelaars de omgeving bepalen waarin een toepassing wordt uitgevoerd. Met de verschillende eigenschappen van de klasse Capabilities kunt u de resolutie van het systeem van de gebruiker, de taal van het besturingssysteem van de gebruiker en de momenteel geïnstalleerde versie van Flash-runtime opvragen en vaststellen of het systeem van de gebruiker toegankelijkheidssoftware ondersteunt.

Door de eigenschappen in de klasse Capabilities te controleren, kunt u de toepassing optimaal aanpassen aan de omgeving van de specifieke gebruiker. Door de eigenschappen `Capabilities.screenResolutionX` en `Capabilities.screenResolutionY` te controleren kunt u bijvoorbeeld bepalen welke resolutie het beeldscherm van het systeem van de gebruiker heeft en welke videogrootte het meest geschikt is. U kunt ook de eigenschap `Capabilities.hasMP3` controleren om na te gaan of het systeem van de gebruiker het afspelen van MP3-bestanden ondersteunt voordat u probeert een extern MP3-bestand te laden.

De volgende code maakt gebruik van een reguliere expressie om de versie van Flash-runtime die de client gebruikt, te parseren:

```
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.");
}
```

U kunt de systeemmogelijkheden van het systeem van de gebruiker met de volgende ActionScript-code naar een serverscript sturen, zodat de informatie kan worden opgeslagen in een database:

```
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);
```

Voorbeeld van mogelijkheden: systeemmogelijkheden bepalen

Flash Player 9 of hoger

Het voorbeeld CapabilitiesExplorer toont hoe u met de klasse flash.system.Capabilities kunt bepalen welke eigenschappen door de Flash-runtimeversie van de gebruiker worden ondersteund. In dit voorbeeld worden de volgende technieken toegepast:

- Bepalen welke mogelijkheden door de Flash-runtimeversie van de gebruiker worden ondersteund met de klasse Capabilities.
- Bepalen welke browserinstellingen door de browser van de gebruiker worden ondersteund met de klasse ExternalInterface.

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van CapabilitiesExplorer vindt u in de map Samples/CapabilitiesExplorer. Deze toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
CapabilitiesExplorer.fla of CapabilitiesExplorer.mxml	Het hoofdtoepassingsbestand in Flash (FLA) of Flex (MXML).
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	De klasse die de hoofdfunctionaliteit van de toepassing biedt, zoals het toevoegen van systeemmogelijkheden aan een array, het sorteren van items en het ophalen van de browsermogelijkheden met de klasse ExternalInterface.
capabilities.html	Een HTML-container die het JavaScript bevat dat is vereist voor communicatie met de externe API.

Overzicht CapabilitiesExplorer

Flash Player 9 of hoger

Het bestand CapabilitiesExplorer.mxml stelt de gebruikersinterface in voor de toepassing CapabilitiesExplorer. De mogelijkheden van de Flash-runtimeversie van de gebruiker worden weergegeven in een componentinstantie DataGrid in het werkgebied. De browsermogelijkheden worden ook weergegeven als de toepassing wordt uitgevoerd vanuit een HTML-container en als de externe API beschikbaar is.

Wanneer de gebeurtenis `creationComplete` van het hoofdtoepassingsbestand is verzonden, wordt de methode `initApp()` aangeroepen. De methode `initApp()` roept de methode `getCapabilities()` aan vanuit de klasse com.example.programmingas3.capabilities.CapabilitiesGrabber. De code voor de methode `initApp()` ziet er als volgt uit:

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

De methode `CapabilitiesGrabber.getCapabilities()` retourneert een gesorteerde array van de mogelijkheden van Flash-runtime en de browser, die vervolgens worden ingesteld op de eigenschap `dataProvider` van de DataGrid-componentinstantie `capabilitiesGrid` in het werkgebied.

Overzicht van de klasse CapabilitiesGrabber

Flash Player 9 of hoger

De statische methode `getCapabilities()` van de klasse CapabilitiesGrabber voegt elke eigenschap van de klasse flash.system.Capabilities toe aan een array (`capDP`). Vervolgens roept de methode de statische methode `getBrowserObjects()` in de klasse CapabilitiesGrabber aan. De methode `getBrowserObjects()` gebruikt de externe API om het navigatorobject van de browser dat de mogelijkheden van de browser bevat, met een lus te doorlopen. De methode `getCapabilities()` ziet er als volgt uit:

```
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;
}
```

De methode `getBrowserObjects()` retourneert een array van alle eigenschappen in het navigatorobject van de browser. Als deze array een lengte heeft van een of meer items, wordt de array van browsermogelijkheden (`navArr`) toegevoegd aan de array van Flash Player-mogelijkheden (`capDP`) en wordt de hele array alfabetisch gesorteerd. Ten slotte wordt de gesorteerde array geretourneerd aan het hoofdtoepassingsbestand. Hiermee wordt vervolgens het gegevensraster gevuld. De code voor de methode `getBrowserObjects()` ziet er als volgt uit:

```
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;
}
```

Als de externe API beschikbaar is in de huidige gebruikersomgeving, roept Flash-runtime de JavaScript-methode `JS_getBrowserObjects()` aan, die het navigatorobject van de browser met een lus doorloopt en een tekenreeks van URL-gecodeerde waarden aan ActionScript retourneert. Deze tekenreeks wordt vervolgens omgezet in een object URLVariables (`itemVars`) en toegevoegd aan de array `itemArr`, die wordt geretourneerd aan het aanroepende script.

Communiceren met JavaScript

Flash Player 9 of hoger

Het sluitstuk van het bouwen van de toepassing CapabilitiesExplorer is het schrijven van het JavaScript dat nodig is om elk item in het navigator-object van de browser met een lus te doorlopen en een naam/waarde-paar toe te voegen aan een tijdelijke array. De code voor de JavaScript-methode `JS_getBrowserObjects()` in het bestand container.html ziet er als volgt uit:

```
<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>
```

De code maakt eerst een tijdelijke array, die alle naam/waarde-paren in het navigatorobject bevat. Vervolgens wordt het navigatorobject doorlopen met een lus `for..in` en wordt het gegevenstype van de huidige waarde geëvalueerd om ongewenste waarden uit te filteren. In deze toepassing zijn we alleen geïnteresseerd in tekenreekswaarden of Booleaanse waarden. Andere gegevenstypen (zoals functies en arrays) worden genegeerd. Elke tekenreekswaarde of Booleaanse waarde in het navigatorobject wordt toegevoegd aan de array `tempArr`. Vervolgens wordt het schermobject van de browser doorlopen met een lus `for..in` en wordt elke numerieke waarde toegevoegd aan de array `tempArr`. Ten slotte wordt de tijdelijke array omgezet in een tekenreeks met de methode `Array.join()`. In deze array wordt de ampersand (&) gebruikt als scheidingsteken, zodat ActionScript de gegevens gemakkelijk kan parseren met de klasse URLVariables.

Hoofdstuk 50: AIR-toepassingen oproepen en beëindigen

Adobe AIR 1.0 of hoger

In deze sectie wordt beschreven hoe de geïnstalleerde Adobe® AIR®-toepassing kan worden aangeroepen, alsook opties en overwegingen voor het sluiten van een actieve toepassing.

Opmerking: *De NativeApplication-, InvokeEvent- en BrowserInvokeEvent-objecten zijn alleen beschikbaar voor actieve SWF-inhoud in de AIR-toepassingssandbox. Actieve SWF-inhoud in de Flash Player-runtime in de browser of aparte player (projector) of in een AIR-toepassing buiten de toepassingssandbox heeft geen toegang tot deze klassen.*

Lees voor een snelle uitleg en codevoorbeelden van het oproepen en beëindigen van AIR-toepassingen, de volgende snelstartartikelen in de Adobe Developer Connection:

- Opstartopties

Meer Help-onderwerpen

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

Toepassingen oproepen

Adobe AIR 1.0 of hoger

Een AIR-toepassing wordt opgeroepen wanneer de gebruiker (of het besturingssysteem):

- de toepassing start vanaf het bureaublad;
- de toepassing gebruikt als een opdracht op een opdrachtregel;
- een type bestand opent waarvoor de toepassing de standaardtoepassing is;
- (Mac OS X) op het toepassingspictogram in de dock-balk klikt (ongeacht of de toepassing is gestart);
- de toepassing start vanuit het installatieprogramma (aan het einde van een nieuw installatieproces of na het dubbelklikken op het AIR-bestand voor een geïnstalleerde toepassing);
- Begint een update van een AIR-toepassing wanneer de geïnstalleerde versie heeft aangegeven dat deze zelf toepassingsupdates verwerkt (door een declaratie `<customUpdateUI>true</customUpdateUI>` op te nemen in het descriptorbestand van de toepassing).
- (iOS) Ontvangt een bericht van de APN's (Apple Push Notification service).
- Roept de toepassing via een URL aan.

- een webpagina bezoekt waarop zich een Flash-badge of -toepassing bevindt die de methode `com.adobe.air.AIR launchApplication()` oproept met de identificatiegegevens voor de AIR-toepassing. (Oproepen vanuit de browser is alleen mogelijk als in de toepassingsdescriptor ook de declaratie `<allowBrowserInvocation>true</allowBrowserInvocation>` is opgenomen.)

Wanneer een AIR-toepassing wordt opgeroepen, wordt een InvokeEvent-object van het type `invoke` verzonden via het NativeApplication-singletonobject. Om een toepassing tijd te geven zichzelf te starten en een gebeurtenislistener te registreren, worden gebeurtenissen van het type `invoke` in de wachtrij geplaatst in plaats van verzonden. Zodra een listener is geregistreerd, worden alle gebeurtenissen in de wachtrij afgeleverd.

Opmerking: *Wanneer een toepassing wordt opgeroepen met de browseroproepfunctie, verzendt het NativeApplication-object de gebeurtenis van het type `invoke` alleen als de toepassing nog niet wordt uitgevoerd.*

Roep de methode `addEventListener()` van het NativeApplication-object (`NativeApplication.nativeApplication`) op om gebeurtenissen van het type `invoke` te ontvangen. Wanneer een gebeurtenislistener een gebeurtenis van het type `invoke` heeft geregistreerd, ontvangt deze ook alle gebeurtenissen van het type `invoke` die vóór de registratie zijn opgetreden. Gebeurtenissen van het type `invoke` in de wachtrij worden een voor een, kort na elkaar verzonden nadat het oproepen van `addEventListener()` is uitgevoerd. Als er een nieuwe gebeurtenis van het type `invoke` optreedt tijdens dit proces, kan deze worden verzonden voordat een of meer gebeurtenissen in de wachtrij zijn verzonden. Omdat gebeurtenissen in een wachtrij worden opgeslagen, kunt u alle opgetreden gebeurtenissen van het type `invoke` verwerken voordat uw initialisatiecode wordt uitgevoerd. Denk eraan dat als u later tijdens de uitvoering (na het initialiseren van de toepassing) een gebeurtenislistener toevoegt, deze nog steeds alle gebeurtenissen van het type `invoke` ontvangt die zijn opgetreden sinds de toepassing is gestart.

Er wordt slechts één instantie van een AIR-toepassing gestart. Wanneer een toepassing die al is gestart opnieuw wordt opgeroepen, wordt een nieuwe gebeurtenis van het type `invoke` verzonden naar de gestarte instantie. Het is de verantwoordelijkheid van een AIR-toepassing om te reageren op een gebeurtenis van het type `invoke` en de juiste actie uit te voeren (zoals het openen van een nieuw documentvenster).

Een `InvokeEvent`-object bevat alle argumenten die aan de toepassing worden doorgegeven en ook de map van waaruit de toepassing is opgeroepen. Als de toepassing is opgeroepen via een gekoppeld bestandstype, wordt het volledige pad naar het bestand opgenomen in de opdrachtregelargumenten. Als de toepassing is opgeroepen via een toepassingsupdate, wordt het volledige pad naar het AIR-updatebestand opgenomen.

Wanneer meer bestanden in één bewerking worden geopend, wordt een enkel InvokeEvent-object verzonden op Mac OS X. Elk bestand is opgenomen in de array `arguments`. In Windows en Linux wordt voor elk bestand een afzonderlijk InvokeEvent-object verzonden.

Uw toepassing kan gebeurtenissen van het type `invoke` verwerken door een listener te registreren met het NativeApplication-object:

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

En een gebeurtenislistener te definiëren:

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

Opdrachtregelargumenten vastleggen

Adobe AIR 1.0 of hoger

De opdrachtregelargumenten die gekoppeld zijn aan de aanroeping van een AIR-toepassing worden geleverd in het InvokeEvent-object dat wordt verzonden door het NativeApplication-object. De eigenschap InvokeEvent-`arguments` bevat een array van de argumenten die door het besturingssysteem worden doorgegeven wanneer er een AIR-toepassing wordt aangeroepen. Als de argumenten relatieve bestandspaden bevatten, kunt u de paden gewoonlijk omzetten met de eigenschap `currentDirectory`.

De argumenten die aan een AIR-programma worden doorgegeven, worden behandeld als met spaties gescheiden tekenreeksen, tenzij ze tussen dubbele aanhalingstekens staan:

Argumenten	Array
tik tak	{tik,tak}
tik "tik tak"	{tik,tik tak}
"tik" “tak”	{tik,tak}
\"tik\" \"tak\"	{"tik","tak"}

De eigenschap `currentDirectory` van een InvokeEvent-object bevat een Bestandsobject die de map weergeeft waarin een toepassing werd gestart.

Wanneer een toepassing wordt opgeroepen omdat een bestandstype wordt geopend dat door de toepassing is geregistreerd, wordt het oorspronkelijke pad naar het bestand als een tekenreeks opgenomen in de opdrachtregelargumenten. (Uw toepassing is verantwoordelijk voor het openen van het bestand of het uitvoeren van de gewenste bewerking.) Wanneer een toepassing is geprogrammeerd om zichzelf te updaten (en niet afhankelijk is van de standaard gebruikersinterface voor AIR-updates), wordt het oorspronkelijke pad naar het AIR-bestand opgenomen wanneer een gebruiker dubbelklikt op een AIR-bestand dat een toepassing met een overeenkomende toepassings-id bevat.

U kunt het bestand openen met de methode `resolve()` van het File-object `currentDirectory`:

```
if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

U moet ook controleren of een argument inderdaad een pad naar een bestand is.

Voorbeeld: logboek met oproepgebeurtenissen

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u listeners voor de `invoke`-gebeurtenis registreert en deze verwerkt. In het voorbeeld worden alle ontvangen oproepgebeurtenissen in het logboek geregistreerd en worden de huidige map en opdrachtregelargumenten weergegeven.

ActionScript-voorbeeld

```
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);
        }
    }
}
```

Flex-voorbeeld

```
<?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>
```

Een AIR-toepassing vanaf aanmeldingsnaam aanroepen

Adobe AIR 1.0 of hoger

Een AIR-toepassing kan worden ingesteld zodat deze automatisch opstart wanneer de huidige gebruiker zich aanmeldt door de eigenschap NativeApplication `startAtLogin` in te stellen op `true`. Wanneer dit is ingesteld, wordt de toepassing automatisch gestart wanneer de gebruiker zich aanmeldt. De toepassing wordt altijd gestart bij het aanmelden, totdat de instelling wordt gewijzigd in `false`, als de gebruiker de instelling handmatig via het besturingssysteem wijzigt of wanneer de toepassing wordt verwijderd. Starten bij aanmelden is een runtime-instelling. De instelling geldt alleen voor de huidige gebruiker. De toepassing moet zijn geïnstalleerd om de eigenschap `startAtLogin` op `true` in te kunnen stellen. Er wordt een fout gemeld als de eigenschap wordt ingesteld wanneer een toepassing niet is geïnstalleerd (bijvoorbeeld als deze wordt gestart met ADL).

Opmerking: *De toepassing wordt niet gestart wanneer de computer wordt opgestart. De toepassing wordt gestart wanneer de gebruiker zich aanmeldt.*

Om te bepalen of een toepassing automatisch is gestart of als gevolg van een gebruikersactie, kunt u de eigenschap `reason` van het InvokeEvent-object bekijken. Als de eigenschap gelijk is aan `InvokeEventReason.LOGIN`, wordt de toepassing automatisch gestart. Voor andere aanroeppaden is de eigenschap `reason` als volgt ingesteld:

- `InvokeEventReason.NOTIFICATION` (iOS only) - De toepassing werd aangeroepen via APN's. Zie Pushberichten gebruiken voor meer informatie over APN's.
- `InvokeEventReason.OPEN_URL` - De toepassing werd aangeroepen door een andere toepassing op door het systeem.
- `InvokeEventReason.Standard` - Alle overige gevallen.

Om de eigenschap `reason` te openen, moet uw toepassing gericht zijn op AIR 1.5.1 of hoger (door de correcte naamruimtewaarde in te stellen in het descriptorbestand van de toepassing).

De volgende, vereenvoudigde toepassing gebruikt de redeneigenschap InvokeEvent om het gedrag te beslissen wanneer er een aanroepgebeurtenis plaatsvindt. Als de redeneigenschap 'login' is, blijft de toepassing op de achtergrond. Anders wordt de hoofdtoepassing zichtbaar. Een toepassing die dit patroon gebruikt, start over het algemeen bij de aanmelding, zodat deze achtergrondverwerking of gebeurteniscontrole kan uitvoeren en er een venster wordt geopend na een aanroepgebeurtenis die door een gebruiker is geactiveerd.

```
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();
            }
        }
    }
}
```

Opmerking: *Verpak en installeer de toepassing om verschil in gedrag te zien. De eigenschap* `startAtLogin` *kan alleen worden ingesteld voor geïnstalleerde toepassingen.*

Een AIR-toepassing vanaf een browser aanroepen

Adobe AIR 1.0 of hoger

Met de browseroproepfunctie kan een website een geïnstalleerde AIR-toepassing starten vanaf een browser. Oproepen vanuit de browser is alleen toegestaan als `allowBrowserInvocation` op `true` is ingesteld in het descriptorbestand van de toepassing:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Wanneer de toepassing wordt opgeroepen vanuit de browser, verzendt het NativeApplication-object van de toepassing een BrowserInvokeEvent-object.

Als u BrowserInvokeEvent-gebeurtenissen wilt ontvangen, roept u de methode `addEventListener()` van het NativeApplication-object (`NativeApplication.nativeApplication`) op in de AIR-toepassing. Wanneer een gebeurtenislistener een BrowserInvokeEvent-gebeurtenis heeft geregistreerd, ontvangt deze ook alle BrowserInvokeEvent-gebeurtenissen die vóór de registratie zijn opgetreden. Deze gebeurtenissen worden verzonden nadat `addEventListener()` is uitgevoerd, maar niet noodzakelijkerwijs vóór andere BrowserInvokeEvent-gebeurtenissen die na de registratie zijn ontvangen. Zo kunt u BrowserInvokeEvent-gebeurtenissen verwerken die zijn opgetreden voordat uw initialisatiecode is uitgevoerd (bijvoorbeeld wanneer de toepassing is opgeroepen vanuit de browser). Denk eraan dat als u later tijdens de uitvoering (na het initialiseren van de toepassing) een gebeurtenislistener toevoegt, deze nog steeds alle BrowserInvokeEvent-gebeurtenissen ontvangt die zijn opgetreden sinds de toepassing is gestart.

Het BrowserInvokeEvent-object bevat de volgende eigenschappen:

Eigenschap	**Beschrijving**
arguments	Een array van argumenten (tekenreeksen) die worden doorgegeven aan de toepassing.
isHTTPS	Geeft aan of de inhoud in de browser gebruikmaakt van het URL-schema HTTPS (`true`) of niet (`false`).
isUserEvent	Geeft aan of de browseroproep heeft geleid tot een gebruikersgebeurtenis (zoals een muisklik). In AIR 1.0 wordt deze eigenschap altijd ingesteld op `true`; in AIR is een gebruikersgebeurtenis vereist voor de browseroproepfunctie.
sandboxType	Het sandboxtype voor de inhoud in de browser. Geldige waarden zijn dezelfde waarden die kunnen worden gebruikt in de eigenschap `Security.sandboxType` en dit zijn: • `Security.APPLICATION`: de inhoud bevindt zich in de beveiligingssandbox van de toepassing. • `Security.LOCAL_TRUSTED`: de inhoud bevindt zich in de lokaal-vertrouwde beveiligingssandbox. • `Security.LOCAL_WITH_FILE`: de inhoud bevindt zich in de lokaal-met-bestandssysteem beveiligingssandbox. • `Security.LOCAL_WITH_NETWORK`: de inhoud bevindt zich in de lokaal-met-netwerk beveiligingssandbox. • `Security.REMOTE`: de inhoud bevindt zich in een extern (netwerk)domein.
securityDomain	Het beveiligingsdomein voor de inhoud van de browser, zoals `"www.adobe.com"` of `"www.voorbeeld.org"`. Deze eigenschap wordt alleen ingesteld voor de inhoud van de externe beveiligingssandbox (voor inhoud afkomstig van een netwerkdomein). De eigenschap wordt niet ingesteld voor de inhoud van een lokale of een beveiligingssandbox van de toepassing.

Als u de browseroproepfunctie gebruikt, moet u rekening houden met de gevolgen voor de beveiliging. Wanneer een website een AIR-toepassing start, kan deze via de eigenschap `arguments` van het BrowserInvokeEvent-object gegevens verzenden. Wees voorzichtig wanneer u deze gegevens gebruikt bij gevoelige bewerkingen, zoals API's die bestanden of code laden. Het risico is afhankelijk van wat de toepassing met de gegevens doet. Als u verwacht dat alleen een specifieke website de toepassing aanroept, moet de toepassing de eigenschap `securityDomain` of het BrowserInvokeEvent-object controleren. U kunt er ook voor zorgen dat de website die de toepassing oproept HTTP's gebruikt, die u kunt controleren door de eigenschap `isHTTPS` van het BrowserInvokeEvent-object te bekijken.

De toepassing moet de doorgegeven gegevens valideren. Als een toepassing bijvoorbeeld verwacht dat URL's naar een bepaald domein worden doorgegeven, moet de toepassing controleren of de URL's echt naar dat domein verwijzen. Dit kan voorkomen dat een aanvaller de toepassing ertoe overhaalt vertrouwelijke gegevens te verzenden.

Een toepassing moet geen BrowserInvokeEvent-argumenten gebruiken die naar lokale resources verwijzen. Een toepassing moet bijvoorbeeld geen File-objecten maken die zijn gebaseerd op een pad dat vanuit de browser is doorgegeven. Als wordt verwacht dat externe paden vanuit de browser worden doorgegeven, moet de toepassing ervoor zorgen dat in de paden niet het protocol `file://` wordt gebruikt in plaats van een extern protocol.

Toepassingen beëindigen

Adobe AIR 1.0 of hoger

De snelste manier om een toepassing te sluiten is het aanroepen van de NativeApplication exit()-methode. Dit werkt prima wanneer uw toepassing geen gegevens heeft om op te slaan of externe resources om op te schonen. Als u `exit()` oproept, worden alle vensters gesloten en wordt de toepassing beëindigd. Als vensters of andere componenten van de toepassing echter het beëindigingsproces mogen onderbreken, bijvoorbeeld om essentiële gegevens op te slaan, moet u de juiste waarschuwingsgebeurtenissen verzenden voordat u `exit()` oproept.

Een toepassing kan ook netjes worden afgesloten door een enkel uitvoeringspad op te geven dat onafhankelijk is van de manier waarop het afsluitingsproces is gestart. De gebruiker (of het besturingssysteem) kan het beëindigen van de toepassing op de volgende manieren activeren:

- door het laatste toepassingsvenster te sluiten wanneer `NativeApplication.nativeApplication.autoExit` op `true` is ingesteld;
- door de opdracht Afsluiten van de toepassing te selecteren vanuit het besturingssysteem, bijvoorbeeld wanneer de gebruiker de opdracht Afsluiten kiest in het standaardmenu. (Dit gebeurt alleen in Mac OS. Windows en Linux beschikken niet over een opdracht om de toepassing af te sluiten via de systeeminterface.)
- door de computer af te sluiten.

Wanneer op een van deze manieren een afsluitopdracht wordt overgebracht via het besturingssysteem, verzendt NativeApplication de gebeurtenis `exiting`. Als geen enkele listener de gebeurtenis `exiting` annuleert, worden alle geopende vensters gesloten. Elk venster verzendt de gebeurtenis `closing` en de gebeurtenis `close`. Als een van de vensters de gebeurtenis `closing` annuleert, stopt het afsluitproces.

Als de volgorde waarin vensters worden gesloten belangrijk is voor de toepassing, luistert u naar de gebeurtenis `exiting` van NativeApplication en sluit u de vensters zelf in de juiste volgorde. Dit kan nodig zijn wanneer u bijvoorbeeld een documentvenster met gereedschappaletten hebt. Het kan onhandig zijn, of erger, als het systeem de paletten heeft gesloten, maar de gebruiker besloot om de afsluitopdracht te annuleren om gegevens op te slaan. In Windows ontvangt u de gebeurtenis `exiting` alleen nadat het laatste venster is gesloten (wanneer de eigenschap `autoExit` van het NativeApplication-object is ingesteld op `true`).

Houd u aan de volgende aanbevolen procedures voor het afsluiten van een toepassing, zodat het gedrag consistent is voor alle platformen, of de afsluitopdracht nu wordt geactiveerd via de interface van het besturingssysteem, menuopdrachten of logica in de toepassing:

1 Verzend altijd de gebeurtenis `exiting` via het NativeApplication-object voordat `exit()` wordt opgeroepen in de code van de toepassing en controleer of geen andere component van de toepassing de gebeurtenis annuleert.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2 Luister naar de gebeurtenis `exiting` van de toepassing vanuit het `NativeApplication.nativeApplication`-object en sluit in de handler alle vensters (verzend eerst de gebeurtenis `closing`). Voer alle benodigde opruimtaken uit, zoals het opslaan van toepassingsgegevens of het verwijderen van tijdelijke bestanden, nadat alle vensters zijn gesloten. Gebruik tijdens het opruimen alleen synchrone methoden, zodat u zeker weet dat deze zijn voltooid voordat de toepassing wordt afgesloten.

Als de volgorde waarin de vensters worden gesloten niet belangrijk is, kunt u de array `NativeApplication.nativeApplication.openedWindows` doorlopen en elk venster achtereenvolgens sluiten. Als de volgorde *wel* belangrijk is, moet u zorgen dat de vensters in de juiste volgorde worden gesloten.

```
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. Vensters moeten altijd hun eigen opruimtaken uitvoeren door te luisteren naar hun eigen `closing`-gebeurtenissen.
4. Gebruik slechts één `exiting`-listener in uw toepassing, omdat handlers die eerder zijn opgeroepen niet kunnen weten of volgende handlers de gebeurtenis `exiting` zullen annuleren (en het zou onverstandig zijn om te vertrouwen op de volgorde van uitvoering).

Hoofdstuk 51: Werken met AIR-runtime- en besturingssysteeminformatie

Adobe AIR 1.0 of hoger

Dit hoofdstuk behandelt de manieren waarop een AIR-toepassing systeembestandskoppelingen kan beheren, gebruikersactiviteit kan detecteren en informatie kan ophalen over de Adobe® AIR®-runtime.

Meer Help-onderwerpen

flash.desktop.NativeApplication

Bestandskoppelingen beheren

Adobe AIR 1.0 of hoger

Koppelingen tussen uw toepassing en een bestandstype moeten worden geregistreerd in de toepassingsdescriptor. Tijdens het installatieproces koppelt het installatieprogramma de AIR-toepassing als de standaard openingstoepassing aan elk geregistreerd bestandstype, tenzij er standaard al een andere toepassing is toegewezen. Het installatieproces van de AIR-toepassing wijzigt een bestaande bestandstypekoppeling niet. Om de koppeling van een andere toepassing over te nemen, dient u de methode `NativeApplication.setAsDefaultApplication()` op te roepen tijdens de runtime.

U controleert het best of de verwachte bestandskoppelingen aanwezig zijn wanneer uw toepassing opstart. De reden hiervoor is dat het installatieprogramma van de AIR-toepassing de bestaande bestandskoppelingen niet wijzigt. Bovendien kunnen de bestandskoppelingen op het systeem van de gebruiker altijd veranderen. Als een andere toepassing de huidige bestandskoppeling heeft, is het ook een kwestie van beleefdheid dit aan de gebruiker te vragen voordat een bestaande koppeling wordt overgenomen.

Met de volgende methoden van de klasse NativeApplication kan een toepassing de bestandskoppelingen beheren. Elke methode gebruikt de bestandstype-extensie als parameter:

Methode	Beschrijving
isSetAsDefaultApplication()	Is true (waar) als de AIR-toepassing momenteel aan het opgegeven bestandstype is gekoppeld.
setAsDefaultApplication()	Maakt de koppeling tussen de AIR-toepassing en de openingsactie van het bestandstype.
removeAsDefaultApplication()	Verwijdert de koppeling tussen de AIR-toepassing en het bestandstype.
getDefaultApplication()	Toont het pad van de toepassing die momenteel aan het bestandstype is gekoppeld.

AIR kan alleen koppelingen beheren voor de bestandstypen die oorspronkelijk zijn geregistreerd in de toepassingsdescriptor. U kunt geen informatie ophalen over de koppelingen van een niet-geregistreerd bestandstype, zelfs als een gebruiker handmatig de koppeling heeft gemaakt tussen dat bestandstype en uw toepassing. Als een van de beheermethoden voor bestandskoppelingen wordt opgeroepen met de extensie voor een bestandstype dat niet is geregistreerd in de toepassingsdescriptor, genereert de toepassing een runtime-uitzondering.

De runtimeversie en het patchniveau ophalen

Adobe AIR 1.0 of hoger

Het NativeApplication-object heeft de eigenschap `runtimeVersion`; dit is de versie van de runtime waarin de toepassing wordt uitgevoerd (een tekenreeks, bijvoorbeeld `"1.0.5"`). Het NativeApplication-object heeft ook de eigenschap `runtimePatchLevel`; dit is het patchniveau van de runtime (een getal, bijvoorbeeld 2960). De volgende code maakt gebruik van deze eigenschappen:

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

AIR-mogelijkheden detecteren

Adobe AIR 1.0 of hoger

Voor een bestand dat bij de Adobe AIR-toepassing gebundeld zit, wordt de eigenschap `Security.sandboxType` ingesteld op de waarde die is gedefinieerd door de constante `Security.APPLICATION`. U kunt inhoud (die al dan niet API's bevat die specifiek zijn voor AIR) laden op basis van het feit of het bestand zich in de Adobe AIR-beveiligingssandbox bevindt, zoals geïllustreerd in de volgende code:

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

Alle bronnen die niet samen met de AIR-toepassing zijn geïnstalleerd, worden aan dezelfde beveiligingssandboxen toegewezen die door Adobe® Flash® Player in een webbrowser zouden worden toegewezen. Externe bronnen worden in sandboxen geplaatst op basis van hun brondomeinen, terwijl lokale bronnen in de lokaal-met-netwerk, lokaal-met-bestandssysteem of lokaal-vertrouwde sandbox worden geplaatst.

U kunt nagaan of de statische eigenschap `Capabilities.playerType` is ingesteld op `"Desktop"` om te controleren of inhoud wordt uitgevoerd in de runtime (en niet in Flash Player in een browser).

Zie “Beveiliging in AIR” op pagina 1144 voor meer informatie.

Gebruikersaanwezigheid bijhouden

Adobe AIR 1.0 of hoger

Het NativeApplication-object verzendt twee gebeurtenissen waarmee u kunt controleren of een gebruiker actief een computer gebruikt. Als er geen muis- of toetsenbordactiviteit wordt vastgesteld tijdens de periode die is bepaald met de eigenschap `NativeApplication.idleThreshold`, verzendt NativeApplication de gebeurtenis `userIdle`. De volgende keer dat het toetsenbord of de muis wordt gebruikt, verzendt het NativeApplication-object de gebeurtenis `userPresent`. Het interval `idleThreshold` wordt gemeten in seconden en heeft standaard de waarde 300 (5 minuten). U kunt met de eigenschap `NativeApplication.nativeApplication.lastUserInput` ook het aantal seconden sinds de laatste gebruikersinvoer opvragen.

De volgende code stelt de drempel voor inactiviteit in op 2 minuten en luistert naar de gebeurtenissen `userIdle` en `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");
});
```

Opmerking: *Tussen twee `userPresent`-gebeurtenissen wordt slechts één `userIdle`-gebeurtenis verzonden.*

Hoofdstuk 52: Werken met native AIR-vensters

Adobe AIR 1.0 of hoger

U kunt de klassen die door de Adobe® AIR® API voor native vensters worden geboden, gebruiken om bureaubladvensters te creëren en te beheren.

Basisbeginselen van native vensters in AIR

Adobe AIR 1.0 of hoger

Lees voor een snelle uitleg en codevoorbeelden van het werken met native vensters in AIR de snelstartartikelen in de Adobe Developer Connection:

- Creating a transparent window application (Flex)
- Interacting with a window (Flex)
- Customizing the look and feel of a native window (Flex)
- Launching windows (Flex)
- Creating toast-style windows (Flex)
- Controlling the display order of windows (Flex)
- Creating resizable, non-rectangular windows (Flex)
- Interacting with a window (Flash)
- Customizing the look and feel of a native window (Flash)
- Creating toast-style windows (Flash)
- Controlling the display order of windows (Flash)
- Creating resizable, non-rectangular windows (Flash)

AIR biedt een gebruikersvriendelijke API die op meerdere platforms kan worden gebruikt. Met deze API voor vensters kunnen met Flash®-, Flex™- en HTML-programmeertechnieken native vensters worden gemaakt (dit zijn vensters die eigen zijn aan het besturingssysteem).

Met AIR hebt u een grote vrijheid bij het ontwikkelen van het uiterlijk van uw toepassing. De vensters die u maakt kunnen er uitzien als een standaardbureaubladtoepassing die passen bij de Apple-stijl wanneer ze worden uitgevoerd op de Mac, die voldoen aan de Microsoft-conventies wanneer ze worden uitgevoerd op Windows en die zich aanpassen aan het Linux-vensterbeheer op Linux zonder dat u een regel platformspecifieke code hoeft te gebruiken. U kunt echter ook het uitbreidbare chroom met verwisselbare skins (weergaven) gebruiken die door het Flex-framework wordt geboden. Met dit chroom kunt u uw eigen stijl ontwikkelen, ongeacht het besturingssysteem waarin uw toepassing wordt uitgevoerd. U kunt zelfs uw eigen vensterchroom tekenen met vector- en bitmapillustraties met volledige ondersteuning voor transparantie. Ook alfaovervloeiing wordt ondersteund, waarmee u een gedeeltelijk transparant venster kunt laten overvloeien in de bureaubladachtergrond. Hebt u genoeg van die rechthoekige vensters? Teken eens een rond venster.

Vensters in AIR

Adobe AIR 1.0 of hoger

AIR ondersteunt drie specifieke API's voor het werken met vensters:

- De ActionScript-georiënteerde NativeWindow-klasse biedt de venster-API van het laagste niveau. Gebruik native vensters in toepassingen die in ActionScript en Flash Professional zijn ontworpen. U kunt de klasse NativeWindow uitbreiden om de vensters te specialiseren die in de toepassing worden gebruikt.
- In de HTML-omgeving kunt u de klasse JavaScript Window gebruiken, net als in een browsergebaseerde webtoepassing. Aanroepen naar JavaScript Window-methoden worden doorgestuurd naar het onderliggende native Window-object.
- De Flex-frameworkklassen mx:WindowedApplication en mx:Window bieden een Flex-"wrapper" voor de klasse NativeWindow. De component WindowedApplication vervangt de component Application wanneer u een AIR-toepassing met Flex maakt en moet altijd worden gebruikt als beginvenster in uw Flex-toepassing.

ActionScript-vensters

Wanneer u vensters maakt met de klasse NativeWindow, kunt u rechtstreeks gebruikmaken van het werkgebied en de weergavelijst van Flash Player. Als u visuele objecten aan een NativeWindow wilt toevoegen, voegt u het object toe aan de weergavelijst van werkgebied van het venster of aan een andere weergaveobjectcontainer in het werkgebied.

HTML-vensters

Wanneer u HTML-vensters maakt, gebruikt u HTML, CSS en JavaScript om inhoud weer te geven. Als u een visueel object aan een HTML-venster wilt toevoegen, voegt u die inhoud toe aan het HTML DOM. HTML-vensters vormen een speciale categorie van NativeWindow. De AIR-host definieert in HTML-vensters een `nativeWindow`-eigenschap die toegang biedt tot de onderliggende NativeWindow-instantie. U kunt deze eigenschap gebruiken om toegang te krijgen tot de NativeWindow-eigenschappen, -methoden en -gebeurtenissen die hier worden beschreven.

Opmerking: *Het JavaScript-object Window heeft ook methoden voor het schrijven van scripts voor het venster waarin het object zich bevindt, zoals* `moveTo()` *en* `close()`*. Als er overlappende methoden beschikbaar zijn, gebruikt u de methode die u zelf geschikt vindt.*

Flex-frameworkvensters

Wanneer u vensters maakt met het Flex-framework, gebruikt u gewoonlijk MXML-componenten om het venster te vullen. Als u een Flex-component aan een venster wilt toevoegen, voegt u het componentelement toe aan de MXML-definitie van het venster. U kunt ook ActionScript gebruiken om inhoud dynamisch toe te voegen. De componenten mx:WindowedApplication en mx:Window zijn ontworpen als Flex-containers en kunnen daarom Flex-componenten direct accepteren, terwijl NativeWindow-objecten dat niet kunnen. Indien nodig kunt u met de eigenschap `nativeWindow` via de objecten WindowedApplication en Window toegang krijgen tot de NativeWindow-eigenschappen en -methoden.

Het beginvenster van de toepassing

Het eerste venster van uw toepassing wordt automatisch voor u gemaakt door AIR. AIR stelt de eigenschappen en inhoud van het venster in met de parameters die zijn opgegeven in het element `initialWindow` van het descriptorbestand van de toepassing.

Als de inhoud van de hoofdmap een SWF-bestand is, maakt AIR een NativeWindow-instantie, laadt het dit SWF-bestand en voegt het dit bestand toe aan het werkgebied van het venster. Als de inhoud van de hoofdmap een HTML-bestand is, maakt AIR een HTML-venster en laadt het dit HTML-bestand.

Klassen van native vensters

Adobe AIR 1.0 of hoger

De API voor native vensters bevat de volgende klassen:

Pakket	Klassen
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

Gebeurtenisstromen voor native vensters

Adobe AIR 1.0 of hoger

Native vensters verzenden gebeurtenissen om belanghebbende componenten te melden dat er een belangrijke wijziging zal worden doorgevoerd of al is doorgevoerd. Veel gebeurtenissen die betrekking hebben op vensters, worden in paren verzonden. De eerste gebeurtenis waarschuwt dat er binnenkort een verandering zal plaatsvinden. De tweede gebeurtenis meldt dat de wijziging heeft plaatsgevonden. U kunt een waarschuwingsgebeurtenis annuleren, maar een meldingsgebeurtenis niet. De volgende reeks gebeurtenissen laat zien hoe de gebeurtenisstroom loopt wanneer een gebruiker op de knop Maximaliseren van een venster klikt:

1. Het NativeWindow-object verzendt de gebeurtenis `displayStateChanging`.
2. Als de gebeurtenis niet door geregistreerde listeners wordt geannuleerd, wordt het venster gemaximaliseerd.
3. Het NativeWindow-object verzendt de gebeurtenis `displayStateChange`.

   Daarnaast verzendt het NativeWindow-object ook gebeurtenissen voor gerelateerde wijzigingen in het formaat en de positie van het venster. Het venster verzendt geen waarschuwingsgebeurtenissen voor deze gerelateerde wijzigingen. De gerelateerde wijzigingen zijn:

   a. De gebeurtenis `move` wordt verzonden als de linkerbovenhoek van het vensters is verplaatst als gevolg van het maximaliseren.

   b. De gebeurtenis `resize` wordt verzonden als het vensterformaat is gewijzigd als gevolg van het maximaliseren.

   Het NativeWindow-object verzendt een vergelijkbare reeks gebeurtenissen bij het minimaliseren, sluiten en verplaatsen van vensters en bij het wijzigen of herstellen van het formaat van vensters.

   De waarschuwingsgebeurtenissen worden alleen verzonden wanneer een wijziging is geactiveerd via het vensterchroom of andere door het besturingssysteem gecontroleerde mechanismen. Wanneer u een venstermethode aanroept om het formaat, de positie of de weergavestatus te wijzigen, verzendt het venster alleen een gebeurtenis om de wijziging te melden. U kunt desgewenst een waarschuwingsgebeurtenis verzenden met de methode `dispatchEvent()` van het venster en vervolgens controleren of uw waarschuwingsgebeurtenis is geannuleerd voordat u de wijziging doorvoert.

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de klassen, methoden, eigenschappen en gebeurtenissen van de venster-API.

Eigenschappen die de stijl en het gedrag van een native venster bepalen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De volgende eigenschappen bepalen het basisuiterlijk en -gedrag van een venster:

- `type`
- `systemChrome`
- `transparent`
- `owner`

Wanneer u een venster maakt, stelt u deze eigenschappen in voor het NativeWindowInitOptions-object dat aan de vensterconstructor wordt doorgegeven. AIR leest de eigenschappen voor het beginvenster van de toepassing vanuit de toepassingsdescriptor. (Dit geldt niet voor de eigenschap `type`, die niet in de toepassingsdescriptor kan worden ingesteld en altijd is ingesteld op `normal`.) De eigenschappen kunnen niet worden gewijzigd nadat het venster is gemaakt.

Sommige instellingen van deze eigenschappen sluiten elkaar uit: `systemChrome` kan niet worden ingesteld op `standard` als `transparent` is ingesteld op `true` of als `type` is ingesteld op `lightweight`.

Venstertypen

Adobe AIR 1.0 of hoger

In de venstertypen van AIR zijn de chroom- en zichtbaarheidskenmerken van het oorspronkelijke besturingssysteem gecombineerd. Op die manier zijn er drie functionele typen vensters ontstaan. Gebruik de constanten die in de klasse NativeWindowType zijn gedefinieerd, om in de code te verwijzen naar de naam van de typen. AIR biedt de volgende venstertypen:

Type	Beschrijving
Normaal	Dit is een normaal venster. Normale vensters maken gebruik van het maximale formaat van het chroom en verschijnen op de taakbalk van Windows en in het venstermenu van Mac OS X.
Utiliteit	Dit is een venster met een palet met hulpmiddelen. Utiliteitsvensters maken gebruik van een afgeslankte versie van het systeemchroom en verschijnen niet op de taakbalk van Windows of in het venstermenu van Mac OS X.
Lichtgewicht	Lichtgewichtvensters hebben geen chroom en verschijnen niet op de taakbalk van Windows of in het venstermenu van Mac OS X. Bovendien beschikken lichtgewichtvensters niet over een systeemmenu (Alt+Spatiebalk) in Windows. Lichtgewichtvensters zijn geschikt voor het weergeven van ballonnetjes met mededelingen en voor besturingselementen, zoals keuzelijsten met invoervak die kortstondig een weergavegebied openen. Wanneer het lichtgewichtvenster `type` wordt gebruikt, moet `systemChrome` zijn ingesteld op `none`.

Vensterchroom

Adobe AIR 1.0 of hoger

Het vensterchroom bestaat uit een set besturingselementen waarmee gebruikers een venster in de bureaubladomgeving kunnen manipuleren. Tot de chroomelementen behoren de titelbalk, titelbalkknoppen, vensterrand en formaatgrepen.

Systeemchroom

U kunt de eigenschap `systemChrome` instellen op `standard` of `none`. Kies `standard` voor het systeemchroom als u het venster wilt voorzien van de reeks standaardbesturingselementen die worden gemaakt en vormgegeven door het besturingssysteem van de gebruiker. Kies `none` als u het venster wilt voorzien van uw eigen chroom. Gebruik de constanten die in de klasse NativeWindowSystemChrome zijn gedefinieerd, om in de code te verwijzen naar de instellingen van het systeemchroom.

Het systeemchroom wordt beheerd door het systeem. Uw toepassing heeft niet rechtstreeks toegang tot de besturingselementen zelf, maar kan reageren op de gebeurtenissen die worden verzonden wanneer de besturingselementen worden gebruikt. Wanneer u het standaardchroom gebruikt voor een venster, moet de eigenschap `transparent` zijn ingesteld op `false` en moet de eigenschap `type` zijn ingesteld op `normal` of `utility`.

Flex-chroom

Wanneer u de Flex-component WindowedApplication of Window gebruikt, kan het venster gebruikmaken van het systeemchroom of van het chroom dat door het Flex-framework wordt geleverd. Als u het Flex-chroom wilt gebruiken, stelt u de eigenschap `systemChrome`, die wordt gebruikt om het venster te maken, in op `none`. Wanneer u Flex 4 spark-componenten gebruikt in plaats van mx-componenten, moet u de skinklasse opgeven. Als u dit niet doet, kunt u Flex-chroom niet gebruiken. U kunt de ingebouwde skins gebruiken of uw eigen skins beschikbaar maken. Het volgende voorbeeld laat zien hoe u de skinklasse WindowedApplication van de ingebouwde spark gebruikt om de vensterchroom beschikbaar te maken:

```
<?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>
```

Zie Using Flex 4: About the AIR window containers: Controlling window chrome voor meer informatie.

Aangepast chroom

Wanneer u een venster maakt zonder systeemchroom, moet u uw eigen chroombesturingselementen toevoegen om de interactie tussen een gebruiker en het venster mogelijk te maken en te sturen. U hebt ook de mogelijkheid om transparante, niet-rechthoekige vensters te maken.

U moet de stijl `showFlexChrome` instellen op `false` om het aangepaste chroom met de componenten mx:WindowedApplication of mx:Window te gebruiken. Anders wordt het Flex-chroom zelf aan uw vensters toegevoegd.

Venstertransparantie

Adobe AIR 1.0 of hoger

Als u alfaovervloeiing wilt toestaan, waarbij het venster deels transparant wordt en overvloeit in de bureaubladachtergrond of in andere vensters, stelt u de eigenschap `transparent` van het venster in op `true`. De eigenschap `transparent` moet worden ingesteld voordat het venster wordt gemaakt en kan niet worden gewijzigd als het venster eenmaal is gemaakt.

Een transparant venster heeft geen standaardachtergrond. Elk venstergebied dat geen object bevat dat door de toepassing is getekend, is onzichtbaar. Als voor een weergegeven object de alfa-instelling kleiner is dan één, is alles wat zich onder het object bevindt, gedeeltelijk zichtbaar door het object heen, inclusief eventuele andere weergaveobjecten, andere vensters en het bureaublad.

Transparante vensters zijn handig wanneer u toepassingen wilt maken met randen die onregelmatig van vorm zijn, geleidelijk vervagen of onzichtbaar lijken. Als u echter grote gebieden met alfaovervloeiing wilt renderen, kan dit traag gaan. Gebruik het effect daarom met mate.

Belangrijk: *In Linux kunnen muisgebeurtenissen volledig transparante pixels niet passeren. Vermijd het maken van vensters met grote, volledig transparante gebieden, aangezien u de toegang van de gebruiker tot andere vensters of items op hun bureaublad mogelijk op onzichtbare wijze zou kunnen blokkeren. In Mac OS X en Windows kunnen muisgebeurtenissen volledig transparante pixels niet passeren.*

Transparantie kan niet worden gebruikt voor vensters die gebruikmaken van het systeemchroom. Bovendien wordt SWF- en PDF-inhoud in HTML mogelijk niet weergegeven in transparante vensters. Zie "Belangrijke opmerkingen voor het laden van SWF- of PDF-inhoud in een HTML-pagina" op pagina 1065 voor meer informatie.

De statische eigenschap `NativeWindow.supportsTransparency` geeft aan of venstertransparantie beschikbaar is. Als transparantie niet wordt ondersteund, wordt de toepassing tegen een zwarte achtergrond geplaatst. In deze gevallen worden transparante gebieden van de toepassing dekkend zwart weergegeven. Het kan nuttig zijn een functie te maken waarop kan worden teruggevallen als de eigenschap `false` test. U kunt bijvoorbeeld een waarschuwingsdialoogvenster voor de gebruiker weergeven of een rechthoekige, niet-transparante gebruikersinterface weergeven.

Bedenk dat transparantie altijd wordt ondersteund door Mac- en Windows-besturingssystemen. Voor ondersteuning in Linux-besturingssystemen is een hulpprogramma voor samenstellend vensterbeheer vereist. Zelfs als zo'n hulpprogramma actief is, is transparantie mogelijk niet beschikbaar vanwege de door de gebruiker gekozen weergaveopties of de hardwareconfiguratie.

Transparantie in een MXML-toepassingsvenster

Adobe AIR 1.0 of hoger

De achtergrond van een MXML-venster is standaard ondoorzichtig, zelfs als u bij het maken van het venster kiest voor *transparent*. (Let op het transparantie-effect in de hoeken van het venster.) Als u een transparante achtergrond voor het venster wilt weergeven, stelt u in het opmaakmodel of het element <mx:Style> dat zich in het MXML-bestand van uw toepassing bevindt, een achtergrondkleur en een alfawaarde in. Met de volgende stijldefinitie krijgt de achtergrond bijvoorbeeld een enigszins transparante groene tint:

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

Transparantie in een HTML-toepassingsvenster

Adobe AIR 1.0 of hoger

De achtergrond van HTML-inhoud die in HTML-vensters en HTMLLoader-objecten wordt weergegeven, is standaard ondoorzichtig, zelfs als het venster waarin de inhoud wordt weergegeven, transparant is. Als u de standaardachtergrond wilt uitschakelen die voor HTML-inhoud wordt weergegeven, stelt u de eigenschap `paintsDefaultBackground` in op `false`. In het volgende voorbeeld wordt een HTMLLoader gemaakt en wordt de standaardachtergrond uitgeschakeld:

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

In dit voorbeeld wordt JavaScript gebruikt om de standaardachtergrond van een HTML-venster uit te schakelen:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Als een element in het HTML-document een achtergrondkleur instelt, is de achtergrond van dat element niet transparant. Het instellen van een waarde voor gedeeltelijke transparantie (of matheid) wordt niet ondersteund. U kunt echter een transparante afbeelding in de PNG-indeling als achtergrond voor een pagina of pagina-element gebruiken om een vergelijkbaar visueel effect te krijgen.

Eigendom van vensters

Een venster kan *eigenaar* zijn van een of meerdere vensters. Deze eigendomvensters worden altijd vóór het hoofdvenster weergegeven, worden tegelijk met het hoofdvenster tot pictogram verkleind of hersteld en worden gesloten wanneer het hoofdvenster wordt gesloten. Eigendom van vensters kan niet worden overgedragen aan een ander venster en kan niet worden verwijderd. Een venster kan slechts het eigendom zijn van één hoofdvenster, maar kan eigenaar zijn van een onbeperkt aantal andere vensters.

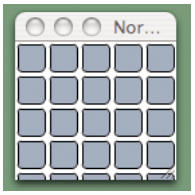
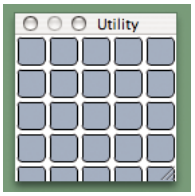
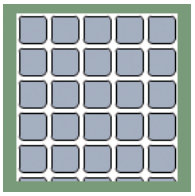
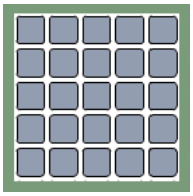
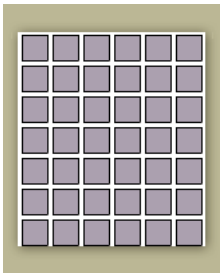
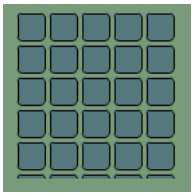
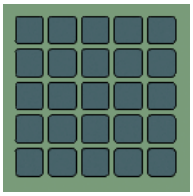
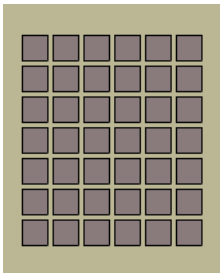
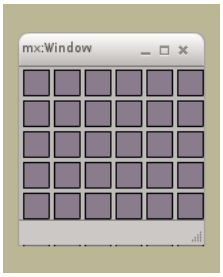
Met venstereigendom kunt u vensters die voor gereedschapspaletten en dialoogvensters worden gebruikt eenvoudiger beheren. Als u bijvoorbeeld een dialoogvenster Opslaan weergeeft in combinatie met een documentvenster en het documentvenster eigenaar maakt van het dialoogvenster, blijft het dialoogvenster automatisch vóór het documentvenster staan.

- NativeWindow.owner
- Christian Cantrell: Vensters in AIR 2.6 die in eigendom zijn

Een visuele venstercatalogus

Adobe AIR 1.0 of hoger

De volgende tabel illustreert de visuele effecten van verschillende combinaties van de instellingen van venstereigenschappen in de besturingssystemen Mac OS X, Windows en Linux:

Vensterinstellingen	Mac OS X	Microsoft Windows	Linux*
Type: normal SystemChrome: standard Transparent: false			
Type: utility SystemChrome: standard Transparent: false			
Type: alle SystemChrome: none Transparent: false			
Type: alle SystemChrome: none Transparent: true			
mxWindowedApplication of mx:Window Type: alle SystemChrome: none Transparent: true			

**Ubuntu met Compiz-vensterbeheer*

Opmerking: *De volgende systeemchroomelementen worden niet ondersteund door AIR: de Mac OS X-werkbalk, het Mac OS X-proxypictogram, pictogrammen op de Windows-titelbalk en het alternatieve systeemchroom.*

Vensters maken

Adobe AIR 1.0 of hoger

AIR maakt automatisch het eerste venster voor een toepassing, maar u kunt zelf alle extra vensters maken die u nodig hebt. Als u een native venster wilt maken, gebruikt u de constructormethode NativeWindow.

Als u een HTML-venster wilt maken, gebruikt u de HTMLLoader-methode `createRootWindow()` of roept u vanuit een HTML-document de JavaScript-methode `window.open()` op. Het venster dat is gemaakt is een NativeWindow-object met een HTMLLoader-object in de weergavelijst. Het HTMLLoader-object interpreteert de HTML- en JavaScript-inhoud voor het venster en geeft deze weer. U hebt toegang tot de eigenschappen van het onderliggende NativeWindow-object vanuit JavaScript met behulp van de eigenschap `window.nativeWindow`. (Deze eigenschap is alleen toegankelijk voor code die wordt uitgevoerd in de sandbox van de AIR-toepassing.)

Wanneer u een venster initialiseert (inclusief het startvenster van de toepassing), zou u het venster eerst in de onzichtbare status kunnen maken, zodat de inhoud wordt geladen en eventuele grafische updates kunnen worden uitgevoerd, voordat u het venster zichtbaar maakt. Door deze volgorde aan te houden voorkomt u dat schokkerige visuele wijzigingen worden weergegeven aan uw gebruikers. U kunt aangeven dat het startvenster van uw toepassing moet worden gemaakt in de onzichtbare status door de tag `<visible>false</visible>` aan te geven in de toepassingsdescriptor (of door de tag weg te laten, aangezien `false` de standaardwaarde is). Nieuwe NativeWindows zijn standaard onzichtbaar. Wanneer u een HTML-venster maakt met de methode HTMLLoader `createRootWindow()`, kunt u het argument `visible` instellen op de waarde `false`. Roep de methode NativeWindow `activate()` aan of stel de eigenschap `visible` in op de waarde `true` om een venster zichtbaar te maken.

Initialisatie-eigenschappen voor vensters opgeven

Adobe AIR 1.0 of hoger

De initialisatie-eigenschappen van een native venster kunnen niet worden gewijzigd nadat het bureaubladvenster is gemaakt. De volgende tabel bevat deze onveranderbare eigenschappen en hun standaardwaarden:

Eigenschap	Standaardwaarde
systemChrome	standard
type	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

U kunt de eigenschappen voor het beginvenster dat door AIR wordt gemaakt, in het descriptorbestand van de toepassing instellen. Het hoofdvenster van een AIR-toepassing is altijd van het type *normal*. (Er kunnen ook extra venstereigenschappen worden opgegeven in het descriptorbestand, zoals `visible`, `width` en `height`, maar deze eigenschappen kunnen op elk gewenst moment worden gewijzigd.)

De eigenschappen voor andere native vensters en HTML-vensters die door uw toepassing zijn gemaakt, kunt u instellen met de klasse NativeWindowInitOptions. Wanneer u een venster maakt, moet u een NativeWindowInitOptions-object waarin de venstereigenschappen zijn opgegeven, doorgeven aan de constructorfunctie NativeWindow of de HTMLLoader-methode `createRootWindow()`.

Met de volgende code maakt u een NativeWindowInitOptions-object voor een utiliteitsvenster:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

U kunt *systemChrome* niet instellen op *standard* wanneer *transparent* is ingesteld op `true` of `type` is ingesteld op `lightweight`.

Opmerking: *U kunt voor een venster dat is gemaakt met de JavaScript-functie `window.open()` geen initialisatie-eigenschappen instellen. U kunt de manier waarop deze vensters zijn gemaakt, echter negeren door uw eigen HTMLHost-klasse te implementeren. Zie "JavaScript-oproepen van window.open() afhandelen" op pagina 1077 voor meer informatie.*

Wanneer u een venster maakt met de Flex-klasse mx:Window, geeft u de initialisatie-eigenschappen op in het vensterobject zelf, in de MXML-declaratie van het venster of in de code waarmee het venster wordt gemaakt. Het onderliggende NativeWindow-object wordt pas gemaakt wanneer u de methode `open()` aanroept. Als een venster eenmaal is geopend, kunnen deze initialisatie-eigenschapen niet meer worden gewijzigd.

Het beginvenster van een toepassing maken

Adobe AIR 1.0 of hoger

AIR maakt het beginvenster van de toepassing op basis van de eigenschappen die zijn opgegeven in de toepassingsdescriptor en laadt het bestand waarnaar wordt verwezen in het element content. Het content-element moet verwijzen naar een SWF- of HTML-bestand.

Het beginvenster kan het hoofdvenster van uw toepassing zijn, maar het kan ook alleen maar worden gebruikt om een of meer andere vensters te starten. U hoeft dit beginvenster niet zichtbaar te maken.

Het beginvenster maken met ActionScript

Adobe AIR 1.0 of hoger

Als u een AIR-toepassing maakt met behulp van ActionScript, moet de hoofdklasse van uw toepassing de Sprite-klasse (of een subklasse van Sprite) uitbreiden. Deze klasse fungeert als primair toegangspunt voor de toepassing.

Wanneer uw toepassing wordt gestart, maakt AIR eerst een venster en vervolgens een hoofdklasse-instantie die aan het werkgebied van het venster wordt toegevoegd. U kunt toegang krijgen tot het venster door te luisteren naar de gebeurtenis `addedToStage` en vervolgens de eigenschap `nativeWindow` van het Stage-object te gebruiken om een verwijzing naar het NativeWindow-object te verkrijgen.

In het volgende voorbeeld wordt het elementaire geraamte getoond voor de hoofdklasse van een AIR-toepassing die met ActionScript is gebouwd:

```
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
        }
    }
}
```

Opmerking: *Technisch gezien KUNT u toegang krijgen tot de eigenschap* `nativeWindow` *in de constructorfunctie van de hoofdklasse. Dit is echter alleen mogelijk in het beginvenster van de toepassing.*

Als u een toepassing in Flash Professional maakt, wordt de hoofddocumentklasse automatisch gemaakt als u er zelf geen hebt gemaakt in een afzonderlijk ActionScript-bestand. U hebt toegang tot het NativeWindow-object voor het beginvenster met behulp van de eigenschap `nativeWindow` van het werkgebied. Met de volgende code bijvoorbeeld kunt u het hoofdvenster op maximumgrootte openen (van de tijdlijn):

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

Het beginvenster maken met Flex

Adobe AIR 1.0 of hoger

Wanneer u een AIR-toepassing maakt met het Flex-framework, moet u mx:WindowedApplication gebruiken als het hoofdelement van uw belangrijkste MXML-bestand. (U kunt de component mx:Application gebruiken, maar deze ondersteunt niet alle functies die beschikbaar zijn in AIR.) De component WindowedApplication fungeert als primair toegangspunt voor de toepassing.

Wanneer u de toepassing start, wordt door AIR een native venster gemaakt, het Flex-framework geïnitialiseerd en het object WindowedApplication toegevoegd aan het werkgebied van het venster. Wanneer de startreeks is voltooid, verzendt de WindowedApplication de gebeurtenis `applicationComplete`. U kunt toegang krijgen tot het bureaubladvensterobject met de eigenschap `nativeWindow` van de WindowedApplication-instantie.

In het volgende voorbeeld wordt een eenvoudige WindowedApplication-component gemaakt waarmee de x- en y-coördinaat van de component worden ingesteld:

```
<?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>
```

NativeWindows maken

Adobe AIR 1.0 of hoger

Als u een NativeWindow wilt maken, geeft u een NativeWindowInitOptions-object door aan de NativeWindow-constructor:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

Het venster wordt pas weergegeven nadat u de eigenschap `visible` op `true` hebt ingesteld of de methode `activate()` hebt aangeroepen.

Als het venster eenmaal is gemaakt, kunt u de eigenschappen van het venster initialiseren en inhoud in het venster laden met de eigenschap Stage en de Flash-technieken voor weergavelijsten.

In vrijwel alle gevallen kunt u de eigenschap `scaleMode` van het werkgebied van een nieuw native venster het best instellen op `noScale` (gebruik de constante `StageScaleMode.NO_SCALE`). De schaalmodi van Flash zijn ontworpen voor situaties waarin de auteur van de toepassing niet van tevoren weet wat de hoogte-breedteverhouding van het weergavegebied voor de toepassing zal zijn. Met de schaalmodi kan de auteur het minst slechte compromis kiezen: de inhoud bijsnijden, uitrekken of samenduwen, of omgeven met lege ruimte. Omdat u de weergaveruimte in AIR (het vensterframe) zelf kunt bepalen, kunt u het formaat van het venster aanpassen aan de inhoud of de inhoud aanpassen aan het venster zonder compromissen te sluiten.

De schaalmodus voor Flex- en HTML-vensters wordt automatisch ingesteld op `noScale`.

Opmerking: *Als u wilt bepalen wat de maximum- en minimumgrootte van een venster is in het huidige besturingssysteem, gebruikt u de volgende statische NativeWindow-eigenschappen:*

```
var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

HTML-vensters maken

Adobe AIR 1.0 of hoger

Als u een HTML-venster wilt maken, kunt u de JavaScript-methode `Window.open()` of de methode `createRootWindow()` van de AIR-klasse HTMLLoader aanroepen.

Voor HTML-inhoud in een willekeurige beveiligingssandbox kan de standaard JavaScript-methode `Window.open()` worden gebruikt. Als de inhoud buiten de toepassingssandbox wordt uitgevoerd, kan de methode `open()` alleen worden aangeroepen als reactie op een handeling van de gebruiker, zoals het klikken met de muis of het indrukken van een toets. Wanneer `open()` wordt aangeroepen, wordt er een venster met het systeemchroom gemaakt waarin de inhoud van de opgegeven URL wordt weergegeven. Bijvoorbeeld:

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

Opmerking: *U kunt de klasse HTMLHost in ActionScript uitbreiden om het venster dat met de JavaScript-functie `window.open()` is gemaakt, naar wens aan te passen. Zie "Informatie over het uitbreiden van de klasse HTMLHost" op pagina 1069.*

Inhoud in de beveiligingssandbox van de toepassing heeft toegang tot de methode `HTMLLoader.createRootWindow()`, die meer mogelijkheden biedt bij het maken van vensters. Met deze methode kunt u alle opties voor het maken van een nieuw venster zelf opgeven. Met de volgende JavaScript-code maakt u bijvoorbeeld een lichtgewichtvenster zonder systeemchroom, dat 300 x 400 pixels groot is:

```
 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"));
```

Opmerking: *als de inhoud die wordt geladen door een nieuw venster, zich buiten de beveiligingssandbox van de toepassing bevindt, zijn de volgende AIR-eigenschappen niet beschikbaar voor het vensterobject: `runtime`, `nativeWindow` en `htmlLoader`.*

Als u een transparant venster maakt, wordt de SWF-inhoud die is ingesloten in de HTML-bestanden opgeslagen en die in dat venster wordt geladen, niet altijd weergegeven. U dient de parameter `wmode` van het object of de insluitmarkering waarmee u naar het SWF-bestand verwijst, instellen op `opaque` of `transparent`. De standaardwaarde van `wmode` is `window` en daarom wordt SWF-inhoud standaard niet weergegeven in transparante vensters. PDF-inhoud kan niet worden weergegeven in transparante vensters, ongeacht de waarde die voor `wmode` wordt ingesteld. (Vóór AIR 1.5.2 kon SWF-inhoud ook niet worden weergegeven in transparante vensters.)

Vensters die zijn gemaakt met de methode `createRootWindow()`, blijven onafhankelijk van het openingsvenster. De eigenschappen `parent` en `opener` van het JavaScript-object Window zijn ingesteld op `null`. Het openingsvenster heeft toegang tot het Window-object van het nieuwe venster met behulp van de HTMLLoader-verwijzing, die is geretourneerd door de functie `createRootWindow()`. In de context van het vorige voorbeeld wordt met de instructie `newHTMLLoader.window` verwezen naar het JavaScript-object Window van het gemaakte venster.

Opmerking: *De functie `createRootWindow()` kan zowel vanuit JavaScript als vanuit ActionScript worden aangeroepen.*

Een mx:Window maken

Adobe AIR 1.0 of hoger

Als u een mx:Window wilt maken, kunt u een MXML-bestand met mx:Window als hoofdtag maken, of u kunt de klasseconstructor Window rechtstreeks aanroepen.

Het volgende voorbeeld maakt en toont een mx:Window door de constructor Window aan te roepen:

```
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);
```

Inhoud toevoegen aan een venster

Adobe AIR 1.0 of hoger

Hoe u inhoud toevoegt aan een AIR-venster hangt af van het type venster. In MXML en HTML bijvoorbeeld kunt u de basisinhoud van het venster declaratief definiëren. U kunt resources insluiten in de SWF-bestanden van de toepassing of u kunt deze laden uit afzonderlijke toepassingsbestanden. Flex-, Flash- en HTML-inhoud kan op elk gewenst moment worden gemaakt en dynamisch aan een venster worden toegevoegd.

Wanneer u SWF-inhoud laadt, of HTML-inhoud die JavaScript bevat, moet u rekening houden met het beveiligingsmodel van AIR. Alle inhoud in de beveiligingssandbox van de toepassing, dat wil zeggen de inhoud die met uw toepassing is geïnstalleerd en kan worden geladen met het schema app: URL, heeft alle benodigde rechten om toegang te krijgen tot alle API's van AIR. Inhoud die van buiten deze sandbox wordt geladen, heeft geen toegang tot de API's van AIR. JavaScript-inhoud buiten de toepassingssandbox kan de eigenschappen `runtime`, `nativeWindow` en `htmlLoader` van het JavaScript-object Window niet gebruiken.

Voor veilige cross-scripting kunt u een sandboxbridge gebruiken om een beperkte interface te bieden tussen toepassingsinhoud en niet-toepassingsinhoud. In HTML-inhoud kunt u ook pagina's van uw toepassing aan een niet-toepassingssandbox toewijzen om de code op die pagina's de mogelijkheid van cross-scripting met externe inhoud te bieden. Zie “Beveiliging in AIR” op pagina 1144.

Een SWF-bestand of -afbeelding laden

Met de klasse `flash.display.Loader` kunt u Flash SWF-bestanden of -afbeeldingen laden in de weergavelijst van een native venster:

```
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);
        }
    }
}
```

Opmerking: *Oudere SWF-bestanden die zijn gemaakt met ActionScript 1 of 2, delen globale statussen, zoals klassedefinities, singletons en globale variabelen, als deze in hetzelfde venster worden geladen. Als een dergelijk SWF-bestand alleen goed kan werken als de globale statussen ongewijzigd zijn, kan het bestand niet meer dan één keer in hetzelfde venster worden geladen. Het bestand kan in dat geval ook niet worden geladen in een venster waarin zich een ander SWF-bestand bevindt dat gebruikmaakt van overlappende klassedefinities en variabelen. Deze inhoud kan in aparte vensters worden geladen.*

HTML-inhoud in een NativeWindow laden

Als u HTML-inhoud in een NativeWindow wilt laden, kunt u een HTMLLoader-object aan het werkgebied van het venster toevoegen en de HTML-inhoud in de HTMLLoader laden, of een venster maken dat al een HTMLLoader-object bevat met de methode `HTMLLoader.createRootWindow()`. In het volgende voorbeeld wordt HTML-inhoud weergegeven binnen een weergavegebied van 300 x 500 pixels op het werkgebied van een native venster:

```
 //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) );
```

Als u een HTML-pagina in een Flex-toepassing wilt laden, kunt u de HTML-component van Flex gebruiken.

SWF-inhoud in een HTML-bestand wordt niet weergegeven als het venster transparantie gebruikt (dat wil zeggen wanneer de eigenschap `transparent` van het venster is ingesteld op `true`), tenzij de parameter `wmode` van het object of de insluitmarkering waarmee naar het SWF-bestand wordt verwezen, is ingesteld op `opaque` of `transparent`. De standaardwaarde voor `wmode` is `window` en daarom wordt SWF-inhoud standaard niet weergegeven in transparante vensters. PDF-inhoud wordt niet weergegeven in transparante vensters, ongeacht de waarde die voor `wmode` wordt gebruikt.

Bovendien wordt noch SWF-, noch PDF-inhoud weergegeven als het besturingselement HTMLLoader is geschaald of geroteerd of als de`alpha`-eigenschap van HTMLLoader is ingesteld op een andere waarde dan 1.0.

SWF-inhoud toevoegen als laag boven op een HTML-venster

Omdat HTML-vensters zich binnen een NativeWindow-instantie bevinden, kunt u zowel boven als onder de HTML-laag in de weergavelijst Flash-weergaveobjecten toevoegen.

Als u een weergaveobject boven de HTML-laag wilt toevoegen, moet u de methode `addChild()` van de eigenschap `window.nativeWindow.stage` gebruiken. Met de methode `addChild()` voegt u inhoud toe als een laag boven alle bestaande inhoud in het venster.

Als u een weergaveobject onder de HTML-laag wilt toevoegen, moet u de methode `addChildAt()` van de eigenschap `window.nativeWindow.stage` gebruiken, waarbij u de waarde nul doorgeeft voor de parameter `index`. Als u een object op de nulindex plaatst, wordt bestaande inhoud, inclusief de HTML-weergave, één laag hoger geplaatst en wordt de nieuwe inhoud helemaal onderaan ingevoegd. Als u wilt dat inhoud op lagen onder de HTML-pagina zichtbaar is, moet u de eigenschap `paintsDefaultBackground` van het object `HTMLlLoader` op `false` instellen. Bovendien zijn elementen op de pagina waarvoor een achtergrondkleur is ingesteld, niet transparant. Als u bijvoorbeeld een achtergrondkleur instelt voor het hoofdtekstelement van de pagina, is geen enkel onderdeel van de pagina transparant.

In het volgende voorbeeld ziet u hoe u Flash-weergaveobjecten als lagen boven en onder de HTML-pagina kunt toevoegen. Met dit voorbeeld maakt u twee eenvoudige vormobjecten, en plaatst u het ene object boven en het andere onder de HTML-inhoud. In dit voorbeeld wordt ook de positie van de vorm bijgewerkt op basis van de gebeurtenis `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>
```

In dit voorbeeld wordt een elementaire inleiding gegeven op bepaalde geavanceerde technieken waarbij de grenzen tussen JavaScript en ActionScript in AIR worden overschreden. Als u niet bekend bent met het gebruik van ActionScript-weergaveobjecten, raadpleegt u “Weergave programmeren” op pagina 161 in de *ActionScript® 3.0-ontwikkelaarsgids*.

Voorbeeld: een native venster maken

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u een native venster kunt maken:

```
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();
}
```

Vensters beheren

Adobe AIR 1.0 of hoger

U kunt de eigenschappen en methoden van de klasse NativeWindow gebruiken om het uiterlijk, het gedrag en de levenscyclus van bureaubladvensters te beheren.

Opmerking: *Als u het Flex-framework gebruikt, is het doorgaans beter venstergedrag te beheren met behulp van de frameworkklassen. U krijgt toegang tot de meeste NativeWindow-eigenschappen en -methoden via de klassen mx:WindowedApplication en mx:Window.*

NativeWindow-instanties opvragen

Adobe AIR 1.0 of hoger

Als u een venster wilt bewerken, moet u eerst een instantie van het venster opvragen. U kunt een vensterinstantie opvragen uit een van de volgende locaties:

- De native vensterconstructor die wordt gebruikt om het venster te maken:

  ```
  var win:NativeWindow = new NativeWindow(initOptions);
  ```

- De eigenschap `nativeWindow` van het werkgebied van het venster:

  ```
  var win:NativeWindow = stage.nativeWindow;
  ```

- De eigenschap `stage`van een weergaveobject in het venster:

  ```
  var win:NativeWindow = displayObject.stage.nativeWindow;
  ```

- De eigenschap `target` van een native venstergebeurtenis die wordt verzonden door het venster:

  ```
  private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
  {
      var win:NativeWindow = event.target as NativeWindow;
  }
  ```

- De eigenschap `nativeWindow` van een HTML-pagina die wordt weergegeven in het venster:

  ```
  var win:NativeWindow = htmlLoader.window.nativeWindow;
  ```

- De eigenschappen `activeWindow` en `openedWindows` van het NativeApplication-object:

  ```
  var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
  var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
  ```

  `NativeApplication.nativeApplication.activeWindow` verwijst naar het actieve venster van een toepassing (maar retourneert de waarde `null` als het actieve venster geen venster van deze AIR-toepassing is). De array `NativeApplication.nativeApplication.openedWindows` bevat alle vensters in een AIR-toepassing die niet zijn gesloten.

Omdat de Flex-objecten mx:WindowedApplication en mx:Window weergaveobjecten zijn, kunt u gemakkelijk naar het toepassingsvenster in een MXML-bestand verwijzen met de eigenschap `stage`. U doet dit als volgt:

```
<?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
```

Opmerking: *Totdat de component WindowedApplication of Window door het Flex-framework wordt toegevoegd aan het werkgebied van het venster, is de eigenschap `stage` van de component ingesteld op `null`. Dit gedrag is consistent met dat van de Flex-component Application. Dit gedrag houdt echter in dat het niet mogelijk is toegang te krijgen tot het werkgebied of de NativeWindow-instantie in listeners voor gebeurtenissen die eerder in de initialisatiecyclus van de componenten WindowedApplication en Window plaatsvinden, zoals `creationComplete`. Het is veilig om het werkgebied en de NativeWindow-instantie te benaderen wanneer de gebeurtenis `applicationComplete` is verzonden.*

Vensters activeren, weergeven en verbergen

Adobe AIR 1.0 of hoger

Als u een venster wilt activeren, roept u de NativeWindow-methode `activate()` op. Als het venster wordt geactiveerd, wordt het venster op de voorgrond weergegeven, krijgt het de focus voor toetsenbord- en muisbewerkingen en wordt het, indien nodig, zichtbaar gemaakt. U maakt het venster zichtbaar door het vorige formaat van het venster te herstellen of de eigenschap `visible` op `true` in te stellen. Als een venster wordt geactiveerd, wordt de volgorde van andere vensters in de toepassing niet gewijzigd. Het aanroepen van de methode `activate()` leidt ertoe dat het venster de gebeurtenis `activate` verzendt.

Als u een verborgen venster wilt weergeven zonder het te activeren, stelt u de eigenschap `visible` in op `true`. Hiermee wordt het venster op de voorgrond geplaatst, maar wordt de focus niet aan het venster toegewezen.

Als u een venster wilt verbergen, stelt u de eigenschap `visible` van dat venster in op `false`. Door een venster te verbergen, wordt de weergave onderdrukt van zowel het venster als de betreffende taakbalkpictogrammen, en - in Mac OS X - ook de toegang tot het Windows-menu.

Wanneer u de zichtbaarheid van een venster wijzigt, wordt de zichtbaarheid van alle vensters waarvan het venster de eigenaar is ook gewijzigd. Als u bijvoorbeeld een venster verbergt, worden alle vensters waarvan het de eigenaar is ook verborgen.

Opmerking: *In Mac OS X is het niet mogelijk om een geminimaliseerd venster met een pictogram in het venstergedeelte van het Dock volledig te verbergen. Als de eigenschap* `visible` *van een geminimaliseerd venster is ingesteld op* `false`*, wordt het Dock-pictogram voor het venster nog wel weergegeven. Als de gebruiker op het pictogram klikt, wordt de zichtbaarheid van het venster hersteld en wordt het venster opnieuw weergegeven.*

De weergavevolgorde van vensters wijzigen

Adobe AIR 1.0 of hoger

AIR biedt verschillende methoden voor het rechtstreeks wijzigen van de weergavevolgorde van vensters. U kunt een venster vooraan of achteraan in de weergavevolgorde plaatsen en u kunt een venster boven of onder een ander venster plaatsen. Tegelijkertijd kan de gebruiker de volgorde van vensters wijzigen door deze te activeren.

U kunt zorgen dat een venster steeds vóór de andere vensters wordt weergegeven door de eigenschap `alwaysInFront` in te stellen op `true`. Als deze instelling is geconfigureerd voor meer dan één venster, bepalen de vensters onderling de weergavevolgorde, maar deze vensters worden wel altijd weergegeven vóór de vensters waarvoor de eigenschap `alwaysInFront` is ingesteld op false.

Vensters in de bovenste groep worden ook weergegeven boven op vensters in andere toepassingen, zelfs wanneer de AIR-toepassing niet actief is. Omdat dit gedrag storend kan zijn voor een gebruiker, moet u de eigenschap `alwaysInFront` alleen op `true` instellen als dat echt nodig en terecht is. Hier volgen enkele voorbeelden van gerechtvaardigd gebruik van deze instelling:

- Tijdelijke pop-upvensters voor besturingen zoals scherminfo, pop-uplijsten, aangepaste menu's of keuzelijsten. Omdat deze vensters gewoonlijk worden gesloten als ze de focus verliezen, wordt voorkomen dat ze de gebruiker het zicht op een ander venster benemen.
- Uiterst urgente foutberichten en waarschuwingen mogen deze instelling hebben. Wanneer er een onomkeerbare wijziging wordt aangebracht als de gebruiker niet tijdig reageert, kan het gerechtvaardigd zijn om een waarschuwingsvenster boven op alle andere vensters weer te geven. De meeste foutberichten en waarschuwingen kunnen echter worden verwerkt in de normale weergavevolgorde van vensters.
- Kortstondig weergegeven, eenvoudige en rechthoekige (toast-stijl) informatievensters mogen deze instelling hebben.

Opmerking: *AIR dwingt een gepast gebruik van de eigenschap* `alwaysInFront` *niet af. Als uw toepassing zich storend gedraagt tijdens de normale werkzaamheden van de gebruiker, zal de toepassing hoogstwaarschijnlijk al snel in de prullenbak van diezelfde gebruiker worden gedeponeerd.*

Als een venster eigenaar is van andere vensters, worden die vensters altijd vóór het eigenaarvenster geplaatst. Als u `orderToFront()` aanroept of `alwaysInFront` instelt op `true` voor een venster dat eigenaar is van andere vensters, worden deze vensters samen met het eigenaarvenster vóór andere vensters gerangschikt, maar worden de eigendomvensters nog steeds vóór de eigenaar weergegeven.

Het aanroepen van methodes voor het rangschikken van vensters voor eigendomvensters werkt op de gebruikelijke wijze voor vensters die eigendom zijn van hetzelfde venster, maar kan ook de volgorde van de volledige groep met eigendomvensters wijzigen in vergelijking met vensters buiten die groep. Als u bijvoorbeeld `orderToFront()` aanroept voor een eigendomvenster, worden dat venster, de eigenaar en alle andere vensters van dezelfde eigenaar vooraan in de vensterrangschikking geplaatst.

De klasse NativeWindow biedt de volgende eigenschappen en methoden voor het instellen van de plaats van een venster ten opzichte van andere vensters in de weergavevolgorde:

Lid	Beschrijving
de eigenschap alwaysInFront	Hiermee geeft u aan of het venster in de bovenste groep vensters moet worden weergegeven. In vrijwel alle gevallen is `false` de beste instelling. Als u de waarde van `false` in `true` wijzigt, wordt het venster boven op alle vensters weergegeven (het venster wordt overigens niet geactiveerd). Als u de waarde van `true` in `false` wijzigt, wordt het venster achter de vensters geplaatst die zich in de bovenste groep bevinden. Het venster wordt wel vóór de andere vensters geplaatst. Als u de eigenschap voor een venster op zijn huidige waarde instelt, wordt de weergavevolgorde van vensters niet gewijzigd. De instelling `alwaysInFront` heeft geen enkele invloed op vensters die het eigendom zijn van een ander venster.
orderToFront()	Hiermee plaatst u het venster op de voorgrond.
orderInFrontOf()	Hiermee plaatst u het venster direct vóór een bepaald venster.
orderToBack()	Hiermee plaatst u het venster achter alle andere vensters.
orderBehind()	Hiermee plaatst u het venster direct achter een bepaald venster.
activate()	Hiermee plaatst u het venster op de voorgrond (en bovendien wordt het venster zichtbaar en wordt de focus toegewezen aan het venster).

Opmerking: *Als een venster verborgen is (`visible` is `false`) of geminimaliseerd, heeft het aanroepen van methoden om de weergavevolgorde te wijzigen, geen effect.*

Bij het Linux-besturingssysteem dwingen verschillende vensterbeheerprogramma's verschillende regels af met betrekking tot de weergavevolgorde van vensters:

- Bij bepaalde vensterbeheerprogramma's worden hulpprogrammavensters altijd weergegeven vóór normale vensters.
- Bij bepaalde vensterbeheerprogramma's wordt een venster dat op het volledige scherm wordt weergegeven en waarvan `alwaysInFront` is ingesteld op `true`, altijd weergegeven vóór andere vensters waarvoor `alwaysInFront` ook is ingesteld op `true`.

Vensters sluiten

Adobe AIR 1.0 of hoger

Als u een venster wilt sluiten, gebruikt u de methode `NativeWindow.close()`.

Als u een venster sluit, wordt de inhoud uit het venster verwijderd, maar als andere objecten verwijzingen naar deze inhoud bevatten, worden de inhoudsobjecten niet vernietigd. De methode `NativeWindow.close()` wordt asynchroon uitgevoerd. Dit betekent dat de toepassing die zich in het venster bevindt, actief blijft tijdens het sluiten van het venster. De methode close verzendt de gebeurtenis close wanneer het sluiten van het venster is voltooid. Het NativeWindow-object functioneert technisch gezien nog steeds, maar als u toegang probeert te krijgen tot eigenschappen en methoden van een gesloten venster, wordt er in de meeste gevallen een IllegalOperationError gegenereerd. U kunt een gesloten venster niet opnieuw openen. Controleer de eigenschap `closed` van een venster om na te gaan of een venster is gesloten. Als u een venster gewoon wilt verbergen, stelt u de eigenschap `NativeWindow.visible` in op `false`.

Als de eigenschap `Nativeapplication.autoExit` op `true` is ingesteld (dit is de standaardinstelling), wordt de toepassing afgesloten op het moment dat het laatste venster wordt gesloten.

Alle vensters met een eigenaar worden gesloten wanneer de eigenaar wordt gesloten. De eigendomvensters verzenden geen sluitgebeurtenis en kunnen het sluiten dus niet voorkomen. Er wordt een sluitgebeurtenis verzonden.

Het annuleren van vensterbewerkingen toestaan

Adobe AIR 1.0 of hoger

Wanneer voor een venster het systeemchroom wordt gebruikt, kunt u de interactie van de gebruiker met het venster annuleren door te luisteren naar de juiste gebeurtenissen en hiervan het standaardgedrag te annuleren. Wanneer een gebruiker bijvoorbeeld op het sluitvak van het systeemchroom klikt, wordt de gebeurtenis `closing` verzonden. Als een geregistreerde listener de methode `preventDefault()` van de gebeurtenis aanroept, wordt het venster niet gesloten.

Wanneer voor een venster niet het systeemchroom wordt gebruikt, worden er ook niet automatisch meldingsgebeurtenissen voor aanstaande wijzigingen verzonden voordat de wijziging wordt aangebracht. Als u daarom de methoden voor het sluiten van een venster aanroept, waarbij de status van het venster wordt gewijzigd, of als u een van de eigenschappen van venstergrenzen instelt, kan de wijziging niet worden geannuleerd. Als u componenten in uw toepassing wilt waarschuwen voordat er een wijziging in een venster wordt aangebracht, kan de logica van uw toepassing de relevante meldingsgebeurtenis verzenden met de methode `dispatchEvent()` van het venster.

Met de volgende logica implementeert u bijvoorbeeld een annuleerbare gebeurtenishandler voor het sluitvak van een venster:

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

De methode `dispatchEvent()` retourneert de waarde `false` als de methode `preventDefault()` van de gebeurtenis door een listener wordt aangeroepen. Aangezien de methode echter ook om andere redenen de waarde `false` kan retourneren, is het beter om expliciet de methode `isDefaultPrevented()` te gebruiken om te controleren of de wijziging moet worden geannuleerd.

Vensters maximaliseren en minimaliseren, of het vorige formaat ervan herstellen

Adobe AIR 1.0 of hoger

Als u het venster wilt maximaliseren, gebruikt u de methode `maximize()` van NativeWindow.

```
myWindow.maximize();
```

Als u het venster wilt minimaliseren, gebruikt u de methode `minimize()` van NativeWindow.

```
myWindow.minimize();
```

Als u het vorige formaat van het venster wilt herstellen (dat wil zeggen het formaat van het venster voordat het werd geminimaliseerd of gemaximaliseerd), gebruikt u de methode `restore()` van NativeWindow.

```
myWindow.restore();
```

Een venster met een eigenaar wordt verkleind tot pictogram en hersteld wanneer het eigenaarvenster wordt verkleind tot pictogram of hersteld. Er worden geen gebeurtenissen verzonden door het eigendomvenster als dit tot pictogram wordt verkleind, aangezien de eigenaar ook tot pictogram is verkleind.

Opmerking: *Het gedrag dat wordt vertoond wanneer u een AIR-venster maximaliseert, verschilt van het standaardgedrag in Mac OS X. Er wordt niet geschakeld tussen het door de toepassing gedefinieerde 'standaardformaat' en het laatste formaat dat door de gebruiker is ingesteld. In plaats daarvan wordt bij een AIR-venster geschakeld tussen het laatste formaat dat door de toepassing is ingesteld en het volledige bruikbare gebied van het scherm.*

Bij het Linux-besturingssysteem dwingen verschillende vensterbeheerprogramma's verschillende regels af met betrekking tot de weergavestatus van vensters:

- Bij bepaalde vensterbeheerprogramma's kunnen vensters van hulpprogramma's niet worden gemaximaliseerd.
- Als een maximumgrootte is ingesteld voor het venster, kunnen in sommige gevallen vensters niet worden gemaximaliseerd. Bepaalde andere vensterbeheerprogramma's stellen de weergavestatus in op gemaximaliseerd, maar wijzigen het formaat van het venster niet. In beide gevallen wordt er geen gebeurtenis betreffende de verandering van de weergavestatus verzonden.
- Bepaalde vensterbeheerprogramma's negeren de vensterinstellingen `maximizable` of `minimizable`.

Opmerking: *Bij Linux worden venstereigenschappen asynchroon gewijzigd. Als u de weergavestatus in één regel van uw programma wijzigt en de waarde in de volgende regel leest, zal de gelezen waarde nog steeds de oude instelling bevatten. Het NativeWindow-object verzendt op alle platforms de gebeurtenis* `displayStateChange` *wanneer de weergavestatus wordt gewijzigd. Als u een actie wilt uitvoeren op basis van de nieuwe status van het venster, moet u dit altijd doen in de gebeurtenishandler* `displayStateChange`*. Zie "Luisteren naar venstergebeurtenissen" op pagina 968.*

Voorbeeld: een venster minimaliseren, maximaliseren, herstellen en sluiten

Adobe AIR 1.0 of hoger

De volgende kleine MXML-toepassing laat zien hoe de methoden `maximize()`, `minimize()`, `restore()` en `close()` van Window werken:

```
<?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>
```

Met het volgende ActionScript-voorbeeld voor Flash worden vier tekstvelden gemaakt waarop kan worden geklikt en waarmee de methoden `minimize()`, `maximize()`, `restore()` en `close()` van NativeWindow kunnen worden geactiveerd:

```
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();
        }
    }
}
```

Het formaat van een venster wijzigen en het venster verplaatsen

Adobe AIR 1.0 of hoger

Wanneer voor een venster het systeemchroom wordt gebruikt, biedt de interface sleepbesturingselementen waarmee u het vensterformaat kunt wijzigen en het venster op het bureaublad kunt verplaatsen. Als voor een venster niet het systeemchroom wordt gebruikt, moet u uw eigen besturingselementen toevoegen om het vensterformaat te wijzigen en het venster te verplaatsen.

Opmerking: *Als u het formaat van een venster wilt wijzigen of een venster wilt verplaatsen, moet u eerst een verwijzing naar de NativeWindow-instantie opvragen. Zie "NativeWindow-instanties opvragen" op pagina 957 voor informatie over het opvragen van een vensterverwijzing.*

Het formaat van een venster wijzigen

Als u een gebruiker wilt toestaan het formaat van een venster interactief te wijzigen, gebruikt u de methode `startResize()` van NativeWindow. Als deze methode wordt aangeroepen vanuit een `mouseDown`-gebeurtenis, wordt het wijzigen van het vensterformaat door de muis gestuurd en is de bewerking voltooid wanneer het besturingssysteem de gebeurtenis `mouseUp` ontvangt. Als u `startResize()` aanroept, geeft u een argument door dat de rand of hoek aangeeft van waaruit het vensterformaat moet worden gewijzigd.

Als u de grootte van het venster wilt programmeren, stelt u de eigenschappen `width`, `height` of `bounds` van het venster in op de gewenste afmetingen. Wanneer u de grenzen instelt, kunt u de grootte en de positie van het venster ook wijzigen. Er is echter geen zekerheid dat deze veranderingen in een bepaalde volgorde zullen worden uitgevoerd. Bepaalde Linux-vensterbeheerprogramma's staan het niet toe dat vensters worden uitgebreid voorbij de randen van het scherm. In dergelijke gevallen is de uiteindelijke venstergrootte vaak beperkt door de volgorde waarin de eigenschappen worden ingesteld, zelfs als het nettoresultaat van de veranderingen een geldige venstergrootte aangeeft. Als u bijvoorbeeld zowel de hoogte als de y-positie van een venster bij de onderkant van het scherm wijzigt, wordt de volledige verandering van de hoogte mogelijk niet doorgevoerd wanneer de hoogte wordt veranderd vóór de verandering van de y-positie.

Opmerking: *Bij Linux worden venstereigenschappen asynchroon gewijzigd. Als u het formaat van een venster in één regel van uw programma wijzigt en de afmetingen in de volgende regel leest, zullen de gelezen afmetingen nog steeds de oude instelling bevatten. Het NativeWindow-object verzendt de gebeurtenis* `resize` *op alle platforms wanneer het formaat van het venster wordt gewijzigd. Als u een actie wilt uitvoeren, zoals het lay-outen van besturingselementen in het venster, op basis van het nieuwe formaat of de nieuwe status van het venster, moet u dit altijd doen in de gebeurtenishandler* `resize`*. Zie "Luisteren naar venstergebeurtenissen" op pagina 968.*

De schaalmodus van het werkgebied bepaalt hoe het werkgebied van het venster en de inhoud daarvan zich gedragen wanneer het formaat van het venster wordt gewijzigd. Houd er rekening mee dat de schaalmodi van het werkgebied zijn ontworpen voor situaties waarin de toepassing de grootte of hoogte-breedteverhouding van zijn weergaveruimte niet zelf kan bepalen (zoals in een webbrowser). U krijgt over het algemeen de beste resultaten door de eigenschap `scaleMode` van het werkgebied in te stellen op `StageScaleMode.NO_SCALE`. Als u wilt dat de inhoud van het venster wordt geschaald, kunt u nog steeds de parameters `scaleX` en `scaleY` van de inhoud instellen als reactie op wijzigingen in de venstergrenzen.

Vensters verplaatsen

Als u een venster wilt verplaatsen zonder het formaat ervan te wijzigen, gebruikt u de methode NativeWindow `startMove()`. Net als voor de methode `startResize()` geldt dat wanneer de methode `startMove()` wordt aangeroepen vanuit de gebeurtenis `mouseDown`, de bewerking (het verplaatsen van het venster) door de muis wordt gestuurd en is voltooid wanneer het besturingssysteem de gebeurtenis `mouseUp` ontvangt.

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de methoden `startResize()` en `startMove()`.

Als u een venster programmatisch wilt verplaatsen, stelt u de eigenschappen `x`, `y` of `bounds` van het venster in op de gewenste positie. Wanneer u de grenzen instelt, kunt u het formaat en de positie van het venster ook wijzigen.

Opmerking: *Bij Linux worden venstereigenschappen asynchroon gewijzigd. Als u een venster in één regel van uw programma verplaatst en de positie in de volgende regel leest, zal de gelezen waarde nog steeds de oude instelling bevatten. Het NativeWindow-object verzendt op alle platforms de gebeurtenis* `move` *wanneer de positie wordt gewijzigd. Als u een actie wilt uitvoeren op basis van de nieuwe positie van het venster, moet u dit altijd doen in de gebeurtenishandler* `move`*. Zie "Luisteren naar venstergebeurtenissen" op pagina 968.*

Voorbeeld: het formaat van vensters wijzigen en vensters verplaatsen

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u bewerkingen zoals het wijzigen van het formaat of het verplaatsen, kunt starten voor een venster:

```
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();
        }
    }
}
```

Luisteren naar venstergebeurtenissen

Adobe AIR 1.0 of hoger

Als u wilt luisteren naar gebeurtenissen die worden verzonden door een venster, registreert u een listener bij de vensterinstantie. Als u bijvoorbeeld naar de gebeurtenis closing wilt luisteren, registreert u als volgt een listener bij het venster:

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

Wanneer een gebeurtenis wordt verzonden, verwijst de eigenschap `target` naar het venster dat de gebeurtenis heeft verzonden.

De meeste venstergebeurtenissen hebben twee gerelateerde berichten. Het eerste bericht geeft aan dat er een vensterwijziging ophanden is (die kan worden geannuleerd), terwijl het tweede bericht aangeeft dat de wijziging heeft plaatsgevonden. Wanneer een gebruiker bijvoorbeeld op het sluitvak van een venster klikt, wordt het bericht van de gebeurtenis closing verzonden. Als de gebeurtenis niet door listeners wordt geannuleerd, wordt het venster gesloten en wordt het bericht van de gebeurtenis close verzonden naar eventuele listeners.

Gewoonlijk worden waarschuwingsgebeurtenissen, zoals `closing`, alleen verzonden wanneer het systeemchroom is gebruikt om een gebeurtenis te activeren. Als bijvoorbeeld de methode `close()` van het venster wordt aangeroepen, wordt niet automatisch de gebeurtenis `closing` verzonden, maar wordt alleen de gebeurtenis `close` verzonden. U kunt echter een closing-gebeurtenisobject maken en dit object verzenden met de methode `dispatchEvent()` van het venster.

De volgende venstergebeurtenissen verzenden een Event-object:

Gebeurtenis	**Beschrijving**
activate	Deze gebeurtenis wordt verzonden wanneer het venster de focus krijgt.
deactivate	Deze gebeurtenis wordt verzonden wanneer het venster de focus verliest.
closing	Deze gebeurtenis wordt verzonden wanneer het venster zal worden gesloten. Dit gebeurt alleen automatisch als op het sluitvak van het systeemchroom wordt geklikt of wanneer in Mac OS X de opdracht Stop wordt aangeroepen.
close	Deze gebeurtenis wordt verzonden wanneer het venster is gesloten.

De volgende venstergebeurtenissen verzenden een NativeWindowBoundsEvent-object:

Gebeurtenis	**Beschrijving**
moving	Deze gebeurtenis wordt verzonden vlak voordat de positie van de linkerbovenhoek van het venster wordt gewijzigd doordat het venster wordt verplaatst of het vensterformaat of de weergavestatus van het venster wordt gewijzigd.
move	Deze gebeurtenis wordt verzonden nadat de positie van de linkerbovenhoek is gewijzigd.
resizing	Deze gebeurtenis wordt verzonden vlak voordat de breedte of hoogte van het venster wordt gewijzigd doordat het vensterformaat of de weergavestatus van het venster wordt gewijzigd.
resize	Deze gebeurtenis wordt verzonden nadat het formaat van het venster is gewijzigd.

Voor NativeWindowBoundsEvent-gebeurtenissen kunt u de eigenschappen `beforeBounds` en `afterBounds` gebruiken om de venstergrenzen te bepalen vóór en na de aanstaande of doorgevoerde wijziging.

De volgende venstergebeurtenissen verzenden een NativeWindowDisplayStateEvent-object:

Gebeurtenis	Beschrijving
displayStateChanging	Deze gebeurtenis wordt verzonden vlak voordat de weergavestatus van het venster wordt gewijzigd.
displayStateChange	Deze gebeurtenis wordt verzonden nadat de weergavestatus van het venster is gewijzigd.

Voor NativeWindowDisplayStateEvent-gebeurtenissen kunt u de eigenschappen `beforeDisplayState` en `afterDisplayState` gebruiken om de weergavestatus van het venster te bepalen vóór en na de aanstaande of doorgevoerde wijziging.

Bij bepaalde Linux-vensterbeheerprogramma's wordt er geen gebeurtenis die een wijziging van de weergavestatus aangeeft verzonden wanneer een venster met een instelling voor maximaal vensterformaat, wordt gemaximaliseerd. (Het venster wordt ingesteld op de maximale weergavestatus, maar het formaat ervan wordt niet gewijzigd.)

Schermvullende vensters weergeven

Adobe AIR 1.0 of hoger

Door de eigenschap `displayState` van het werkgebied in te stellen op `StageDisplayState.FULL_SCREEN_INTERACTIVE` wordt het venster in het volledige scherm weergegeven; in deze modus is toetsenbordinvoer *wel* toegestaan. (In SWF-inhoud die in een browser wordt uitgevoerd, is toetsenbordinvoer niet toegestaan.) Als de gebruiker de schermvullende modus wil verlaten, kan deze op Esc drukken.

Opmerking: *Bepaalde Linux-vensterbeheerprogramma's wijzigen de vensterafmetingen niet zo dat het scherm helemaal wordt gevuld wanneer een maximumformaat is ingesteld voor het venster (het venstersysteemchroom wordt echter wel verwijderd).*

Met de volgende Flex-code wordt bijvoorbeeld een eenvoudige AIR-toepassing gedefinieerd die een eenvoudige schermvullende terminal instelt:

```
 <?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>
```

In het volgende ActionScript-voorbeeld voor Flash wordt een eenvoudige schermvullende teksterminal gesimuleerd:

```
    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);
        }
    }
```

Hoofdstuk 53: Displays in AIR

Adobe AIR 1.0 of hoger

Gebruik de Adobe® AIR®-klasse Screen om informatie op te vragen over de displays die zijn aangesloten op een computer.

Meer Help-onderwerpen

flash.display.Screen

Basisbeginselen van displays in AIR

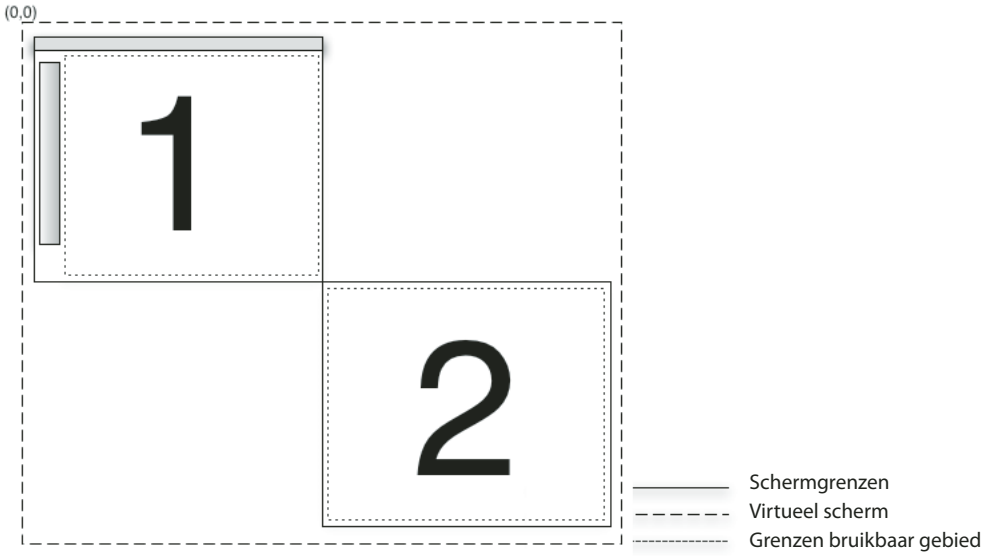
Adobe AIR 1.0 of hoger

- Het virtuele bureaublad meten (Flex)
- het virtuele bureaublad meten (Flash)

De API voor schermen bevat één enkele klasse, Screen, die statische leden gebruikt om scherminformatie op te halen bij het systeem, en instantieleden om een bepaald scherm te beschrijven.

Op een computersysteem kunnen verschillende monitors of displays zijn aangesloten, die kunnen overeenkomen met verschillende bureaubladschermen in een virtuele ruimte. De AIR-klasse Screen geeft informatie over de schermen, hun relatieve opstelling en hun bruikbare ruimte. Als meer dan één monitor aan hetzelfde scherm is toegewezen, bestaat er slechts één scherm. Als een scherm groter is dan het weergavegebied van de monitor, kan niet worden bepaald welk gedeelte van het scherm momenteel zichtbaar is.

Een scherm komt overeen met een onafhankelijk weergavegebied op het bureaublad. Schermen worden gedefinieerd als rechthoeken binnen het virtuele bureaublad. De linkerbovenhoek van het scherm dat als primair display is gedefinieerd, vormt de oorsprong van het coördinatensysteem van het virtuele bureaublad. Alle waarden waarmee een scherm wordt beschreven, worden opgegeven in pixels.



In deze schermopstelling bestaan er twee schermen op het virtuele bureaublad. De coördinaten van de linkerbovenhoek van het hoofdscherm (nr. 1) zijn altijd (0,0). Als de schermopstelling zo verandert dat scherm nr. 2 het hoofdscherm wordt, worden de coördinaten van scherm nr. 1 negatief. Menubalken, taakbalken en docks worden niet meegerekend wanneer de bruikbare grenzen voor een scherm worden opgegeven.

Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor meer informatie over de klassen, methoden, eigenschappen en gebeurtenissen van de scherm-API.

Schermen opsommen

Adobe AIR 1.0 of hoger

U kunt de schermen op het virtuele bureaublad opsommen met de volgende methoden en eigenschappen van Screen:

Methode of eigenschap	Beschrijving
Screen.screens	Geeft een array van Screen-objecten waarmee de beschikbare schermen worden beschreven. De volgorde in de array is niet van belang.
Screen.mainScreen	Geeft een Screen-object voor het hoofdscherm. In Max OS X is het hoofdscherm het scherm waarop de menubalk wordt weergegeven. In Windows is het hoofdscherm het scherm dat door het systeem als primair scherm is aangewezen.
Screen.getScreensForRectangle()	Geeft een array met Screen-objecten die de schermen beschrijven die door de gegeven rechthoek worden gesneden. De rechthoek die aan deze methode wordt doorgegeven, wordt uitgedrukt in pixelcoördinaten op het virtuele bureaublad. Als geen schermen de rechthoek snijden, is de array leeg. U kunt deze methode gebruiken om na te gaan op welke schermen een venster wordt weergegeven.

Sla de waarden niet op die door de methoden en eigenschappen van de klasse Screen worden geretourneerd. De gebruiker of het besturingssysteem kan de beschikbare schermen en hun opstelling op een willekeurig moment wijzigen.

Het volgende voorbeeld gebruikt de API voor schermen om een venster tussen verschillende schermen te verplaatsen wanneer op de pijltoetsen wordt gedrukt. Om het venster naar het volgende scherm te verplaatsen, haalt het voorbeeld de array `screens` op en wordt deze verticaal of horizontaal gesorteerd (afhankelijk van de ingedrukte pijltoets). De code doorloopt vervolgens de gesorteerde array en vergelijkt elk scherm met de coördinaten van het huidige scherm. Om het huidige scherm van het venster te identificeren, roept het voorbeeld `Screen.getScreensForRectangle()` op door de venstergrenzen door te geven.

```
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;
        }
    }
}
```

Hoofdstuk 54: Afdrukken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Adobe® Flash® Player en Adobe ® AIR™ kunnen met de afdrukinterface van een besturingssysteem communiceren zodat u pagina's kunt doorgeven aan de afdrukspooler. Elke pagina die naar de spooler is verzonden, kan inhoud bevatten die voor de gebruiker zichtbaar, dynamisch of buiten beeld is, met inbegrip van databasewaarden en dynamische tekst. Bovendien bevatten de eigenschappen van de klasse flash.printing.PrintJob waarden die zijn ingesteld op basis van de printerinstellingen van de gebruiker, zodat u pagina's correct kunt opmaken.

In dit hoofdstuk vindt u informatie over strategieën voor het gebruik van de methoden en eigenschappen van de klasse flash.printing.PrintJob voor het maken van een afdruktaak, het lezen van afdrukinstellingen van gebruikers en het aanpassen van een afdruktaak op basis van feedback van de Flash-runtime en het besturingssysteem van de gebruiker.

Basisbeginselen van afdrukken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 3.0 gebruikt u de klasse PrintJob voor het maken van momentopnamen van weergave-inhoud die vervolgens op papier kunnen worden afgedrukt. In sommige opzichten is het instellen van inhoud voor afdrukken hetzelfde als het instellen hiervan voor weergave op het scherm (u plaatst elementen en wijzigt de grootte van de elementen om de gewenste lay-out te maken). Bij afdrukken zijn er echter een aantal bijzondere eigenschappen waardoor deze anders is dan de schermlay-out. Printers gebruiken bijvoorbeeld een andere resolutie dan computerschermen. De inhoud van een computerscherm is dynamisch en kan worden gewijzigd, terwijl afgedrukte inhoud statisch is. Ook moet u bij het maken van afdrukken rekening houden met de beperkingen van een vaste paginagrootte en de mogelijkheid voor het afdrukken van meerdere pagina's.

Deze verschillen lijken misschien vanzelfsprekend, maar u moet hier rekening mee houden wanneer u instellingen opgeeft voor het maken van afdrukken via ActionScript. Accuraat afdrukken is afhankelijk van een combinatie van de door u opgegeven waarden en de eigenschappen van de printer van de gebruiker. De klasse PrintJob bevat eigenschappen waarmee u de belangrijkste eigenschappen van de printer van de gebruiker kunt definiëren.

Belangrijke concepten en termen

De volgende referentielijst bevat belangrijke termen met betrekking tot afdrukken:

Spooler Het gedeelte van het besturingssysteem of software van het printerstuurprogramma dat het aantal pagina's bijhoudt dat in de wacht staat om te worden afgedrukt en ze naar de printer verzendt wanneer deze beschikbaar is.

Afdrukstand De rotatie van de afgedrukte inhoud ten opzichte van het papier. Dit is horizontaal (liggend) of verticaal (staand).

Afdruktaak De pagina of set pagina's die een enkele afdruk vormen.

Pagina afdrukken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U gebruikt een instantie van de klasse PrintJob om het afdrukken af te handelen. Voor het afdrukken van een basispagina via Flash Player of AIR, gebruikt u de volgende vier instructies in de aangegeven volgorde:

- `new PrintJob()`: hiermee wordt een nieuwe instantie van een afdruktaak gemaakt met de door u opgegeven naam.
- `PrintJob.start()`: hiermee wordt het afdrukproces voor het besturingssysteem gestart, waarbij het afdrukdialoogvenster voor de gebruiker wordt aangeroepen en de alleen-lezen eigenschappen van de afdruktaak worden gevuld.
- `PrintJob.addPage()`: bevat de gegevens van de inhoud van de afdruktaak, met inbegrip van het object Sprite (en alle daarin opgenomen onderliggende items), de grootte van het afdrukgebied en informatie over het feit of de printer de afbeelding als een vector of bitmap moet afdrukken. Met opeenvolgende aanroepen van `addPage()` kunt u meerdere Sprites op diverse pagina's afdrukken.
- `PrintJob.send()`: hiermee worden de pagina's verzonden naar de printer van het besturingssysteem.

Hier volgt bijvoorbeeld een eenvoudig script voor een printtaak (inclusief de instructies `package`, `import` en `class` voor het compileren):

```
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();
        }
    }
}
```

Opmerking: *In dit voorbeeld worden alleen de basiselementen getoond van een script voor een afdruktaak. Er is geen foutafhandeling opgenomen. Zie "Werken met uitzonderingen en geretourneerde meldingen" op pagina 979 als u een script wilt bouwen dat correct reageert op het annuleren van een afdruktaak door een gebruiker.*

Als u de eigenschappen van een PrintJob-object wilt wissen, stelt u de PrintJob-variabele op `null` in (zoals in `myPrintJob = null`).

Taken van de Flash-runtime en afdrukken via het besturingssysteem

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat de Flash-runtime pagina's verzendt naar de afdrukinterface van het besturingssysteem, is het van belang dat u weet welke taken worden beheerd door de Flash-runtime en welke door de eigen afdrukinterface van een besturingssysteem. De Flash-runtime kan een afdruktaak starten, enkele pagina-instellingen van een printer lezen, de inhoud voor een afdruktaak doorgeven aan het besturingssysteem en controleren of de gebruiker dan wel het systeem een afdruktaak heeft geannuleerd. Andere processen, zoals het weergeven van printerspecifieke dialoogvensters, het annuleren van een gespoolde afdruktaak of het melden van de printerstatus, worden allemaal door het besturingssysteem afgehandeld. De Flash-runtime kan reageren als er een probleem is met het starten of formatteren van een afdruktaak, maar kan alleen meldingen genereren over bepaalde eigenschappen of voorwaarden van de afdrukinterface van het besturingssysteem. Als ontwikkelaar moet u er dan ook voor zorgen dat uw code op deze eigenschappen of voorwaarden kan reageren.

Werken met uitzonderingen en geretourneerde meldingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Ga na of de methode `PrintJob.start()` de waarde `true` retourneert voordat u de aanroepen `addPage()` en `send()` uitvoert, voor het geval de gebruiker de afdruktaak heeft geannuleerd. Een eenvoudige manier om voordat u verdergaat te controleren of deze methoden zijn geannuleerd, is ze als volgt op te nemen in een instructie `if`:

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

Als `PrintJob.start()` is ingesteld op `true`, heeft de gebruiker de opdracht Afdrukken geselecteerd (of is deze opdracht geactiveerd door de Flash-runtime (bijvoorbeeld door Flash Player of AIR)). De methoden `addPage()` en `send()` kunnen dus worden aangeroepen.

Voor het beheren van het afdrukproces, genereren Flash Player en AIR bovendien uitzonderingen voor de methode `PrintJob.addPage()`, zodat u fouten kunt afvangen en de gebruiker van informatie en opties kunt voorzien. Als een methode `PrintJob.addPage()` mislukt, kunt u ook een andere functie aanroepen of de huidige afdruktaak stopzetten. U vangt deze uitzonderingen af door aanroepen `addPage()` in te sluiten in een instructie `try..catch`, zoals in het volgende voorbeeld. In het voorbeeld is `[params]` een plaatsaanduiding voor de parameters die aangeven welke inhoud u wilt afdrukken:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

Nadat de afdruktaak is begonnen, kunt u de inhoud toevoegen met behulp van `PrintJob.addPage()` en controleert u vervolgens of daardoor een uitzondering wordt gegenereerd (bijvoorbeeld als de gebruiker de afdruktaak heeft geannuleerd). Als dit geval het geval is, kunt u logica toevoegen aan de instructie `catch` om de gebruiker (of de Flash runtime) te voorzien van informatie en opties. Als de pagina wordt toegevoegd, kunt u meerdere pagina's naar de printer sturen met `PrintJob.send()`.

Als de Flash runtime een probleem ondervindt tijdens het verzenden van de afdruktaak naar de printer (bijvoorbeeld omdat de printer offline is), kunt u ook die uitzonderingsfout onderscheppen en de gebruiker voorzien van informatie of andere opties (zoals het weergeven van een tekstbericht of het opnemen van een waarschuwing in een animatie). U kunt bijvoorbeeld nieuwe tekst toevoegen aan een tekstveld in een instructie `if..else`, zoals in de volgende code:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

Een werkend voorbeeld vindt u in "Voorbeeld van afdrukken: schalen, bijsnijden en reageren" op pagina 989.

Werken met pagina-eigenschappen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Zodra de gebruiker op OK klikt in het dialoogvenster Afdrukken en `PrintJob.start()` de waarde `true` retourneert, hebt u toegang tot de eigenschappen zoals gedefinieerd door de instellingen van de printer. Tot deze instellingen behoren de hoogte en de breedte van het papier (`pageHeight` en `pageWidth`), evenals de afdrukstand van de inhoud op het papier. Omdat dit printerinstellingen zijn, die niet worden bestuurd door Flash Player of AIR, kunt u deze instellingen niet wijzigen. U kunt ze echter wel gebruiken om de inhoud die u naar de printer stuurt uit te lijnen in overeenstemming met de huidige instellingen. Zie "Formaat, schaal en afdrukstand instellen" op pagina 981 voor meer informatie.

Vector- of bitmaprendering instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de afdruktaak handmatig zodanig instellen dat elke pagina als een vector- of bitmapafbeelding wordt gespoold. In bepaalde gevallen wordt bij vectorafdrukken een kleiner spoolbestand gegenereerd en een beter beeld verkregen dan bij bitmapafdrukken. Als de inhoud echter een bitmapafbeelding bevat en u wilt alfatransparantie- of kleureffecten behouden, moet u de pagina afdrukken als een bitmapafbeelding. Bovendien zet een niet-PostScript-printer de vectorafbeeldingen automatisch om naar bitmapafbeeldingen.

U stelt de afdrukmethode met bitmaps in door een PrintJobOptions-object door te geven als de derde parameter van `PrintJob.addPage()`.

Bij Flash Player en AIR-versies lager dan AIR 2 stelt u de parameter `printAsBitmap` van het PrintJobOptions-object in op `true`, zoals hier:

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

Als u geen waarde voor de derde parameter opgeeft, maakt de afdruktaak gebruik van de standaardinstelling, te weten vectorafdrukken.

Voor AIR 2 en hoger geeft u de afdrukmethode op met de eigenschap `printMethod` van het PrintJobOptions-object. Deze eigenschap accepteert drie waarden die zijn gedefinieerd als constanten in de PrintMethod-klasse:

- `PrintMethod.AUTO`: hiermee wordt automatische de beste afdrukmethode gekozen, op basis van de inhoud die wordt afgedrukt. Als een pagina bijvoorbeeld uit tekst bestaat, wordt de afdrukmethode met vectoren gekozen. Als er echter een watermerkafbeelding met alfatransparantie op de tekst wordt geplaatst, wordt voor de afdrukmethode met bitmaps gekozen, zodat de transparantie intact blijft.
- `PrintMethod.BITMAP`: hiermee wordt de afdrukmethode met bitmaps geforceerd, ongeacht de inhoud
- `PrintMethod.VECTOR`: hiermee wordt de afdrukmethode met vectoren geforceerd, ongeacht de inhoud

Tijdsinstelling bepalen voor instructies voor afdruktaken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In ActionScript 3.0 is een object PrintJob niet beperkt tot één frame (zoals in vorige versies van ActionScript). Maar omdat het besturingssysteem statusinformatie over het afdrukken weergeeft nadat de gebruiker in het dialoogvenster Afdrukken op de knop OK heeft geklikt, moet u `PrintJob.addPage()` en `PrintJob.send()` zo spoedig mogelijk aanroepen om pagina's naar de spooler te verzenden. Een vertraging die optreedt bij het frame dat de aanroep `PrintJob.send()` bevat, zorgt voor een vertraging van het afdrukproces.

In ActionScript 3.0 geldt een time-outlimiet van 15 seconden voor scripts. Daarom kan de tijd tussen elke hoofdinstructie in een reeks afdruktaken niet de limiet van 15 seconden overschrijden. Met andere woorden, de time-outlimiet van 15 seconden voor scripts geldt voor de volgende intervallen:

- Tussen `PrintJob.start()` en de eerste `PrintJob.addPage()`
- Tussen `PrintJob.addPage()` en de volgende `PrintJob.addPage()`
- Tussen de laatste `PrintJob.addPage()` en `PrintJob.send()`

Wanneer een of meer van deze intervallen langer duren dan 15 seconden, retourneert de volgende aanroep `PrintJob.start()` voor de PrintJob-instantie `false` en veroorzaakt de volgende `PrintJob.addPage()` voor de PrintJob-instantie een runtime-uitzondering in Flash Player of AIR.

Formaat, schaal en afdrukstand instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de sectie "Pagina afdrukken" op pagina 978 worden de stappen beschreven voor een elementaire afdruktaak, waarbij de uitvoer rechtstreeks de afgedrukte equivalent is van de schermgrootte en schermpositie van de opgegeven Sprite. Printers gebruiken voor het afdrukken echter verschillende resoluties en kunnen instellingen hebben die een ongunstig effect hebben op de weergave van de afgedrukte Sprite.

Flash runtimes kunnen de afdrukinstellingen van het besturingssysteem lezen, maar deze eigenschappen zijn alleen-lezen: u kunt wel op de waarden reageren, maar u kunt ze niet instellen. U kunt bijvoorbeeld de printerinstellingen voor het paginaformaat opvragen om de inhoud aan dat formaat aan te passen. U kunt ook de marge-instellingen en afdrukstand van de printer opvragen. U kunt op die printerinstellingen reageren door een afdrukgebied op te geven, het verschil tussen de schermresolutie en de puntgrootte van de printer te compenseren of de inhoud aan te passen aan de formaat- of afdrukstandinstellingen van de printer van de gebruiker.

Rechthoeken gebruiken voor het afdrukgebied

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `PrintJob.addPage()` kunt u het gebied opgeven voor een Sprite die u wilt afdrukken. De tweede parameter, `printArea`, wordt opgegeven in de vorm van een object Rectangle. U kunt op drie manieren een waarde voor deze parameter opgeven:

- Maak een object Rectangle met specifieke eigenschappen en gebruik die rechthoek in de aanroep van `addPage()`, zoals getoond in het volgende voorbeeld:

  ```
  private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
  myPrintJob.addPage(sheet, rect1);
  ```

- Als u nog geen object Rectangle hebt opgegeven, kunt u dat alsnog in de aanroep zelf doen, zoals getoond in het volgende voorbeeld:

  ```
  myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
  ```

- Als u waarden wilt opgeven voor de derde parameter in de aanroep van `addPage()`, maar geen rechthoek wilt opgeven, kunt u `null` als tweede parameter gebruiken, zoals getoond in het volgende voorbeeld:

  ```
  myPrintJob.addPage(sheet, null, options);
  ```

Punten en pixels vergelijken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De hoogte en breedte van een rechthoek worden uitgedrukt in pixelwaarden. Een printer werkt met punten als afdrukeenheden. Punten hebben een vast, fysiek formaat (1/72 inch), maar de grootte van een pixel op het scherm is afhankelijk van de resolutie van dat specifieke beeldscherm. Hoe pixels in punten worden omgezet, is afhankelijk van de printerinstellingen en of de Sprite is geschaald. Een niet-geschaalde Sprite die 72 pixels breed is, wordt ongeacht de schermresolutie met een breedte van één inch afgedrukt, waarbij één punt gelijk is aan één pixel.

U kunt de volgende equivalenten gebruiken om inches of centimeters om te zetten in twips of punten (een twip is 1/20 punt):

- 1 punt = 1/72 inch = 20 twips
- 1 inch = 72 punten = 1440 twips
- 1 centimeter = 567 twips

Als u de parameter `printArea` weglaat of als deze onjuist wordt doorgegeven, wordt het volledige gebied van de Sprite afgedrukt.

Schalen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u een object Sprite wilt schalen voordat u dit afdrukt, stelt u de schaaleigenschappen in (zie "Grootte manipuleren en objecten schalen" op pagina 190) voordat u de methode `PrintJob.addPage()` aanroept en stelt u deze eigenschappen na het afdrukken weer in op de oorspronkelijke waarden. De schaal van een object Sprite is niet gekoppeld aan de eigenschap `printArea`. Met andere woorden, als u een afdrukgebied van 50 bij 50 pixels opgeeft, worden er 2500 pixels afgedrukt. Als u het object Sprite schaalt, worden ook 2500 pixels afgedrukt, maar wordt het object met de geschaalde grootte afgedrukt.

Een voorbeeld vindt u in "Voorbeeld van afdrukken: schalen, bijsnijden en reageren" op pagina 989.

Liggend of staand afdrukken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Omdat Flash Player en AIR de instellingen voor de afdrukstand kunnen detecteren, kunt u logica in uw ActionScript opnemen om de grootte of afdrukstand aan te passen op basis van de printerinstellingen, zoals getoond in het voorbeeld:

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

Opmerking: *Als u van plan bent de systeeminstelling te lezen voor de afdrukstand van de inhoud op het papier, moet u eraan denken om de klasse PrintJobOrientation te importeren. De klasse PrintJobOrientation biedt constante waarden die de afdrukstand van de inhoud op de pagina bepalen. U importeert de klasse met de volgende instructie:*

```
import flash.printing.PrintJobOrientation;
```

Reageren op de hoogte en breedte van het papier

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met behulp van een strategie die lijkt op het omgaan met de printerinstellingen voor de afdrukstand, kunt u de pagina-instellingen voor hoogte en breedte lezen en daarop reageren door logica in te sluiten in een instructie `if`. In de volgende code wordt een voorbeeld gegeven:

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

Bovendien kunnen de marge-instellingen van een pagina worden bepaald door de afmetingen van pagina en papier te vergelijken, zoals in het voorbeeld hieronder wordt getoond:

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

Geavanceerde afdruktechnieken

Adobe AIR 2 of hoger

Te beginnen bij Adobe AIR 2 beschikt de PrintJob-klasse over extra eigenschappen en methoden, en worden drie extra klassen ondersteund: PrintUIOptions, PaperSize en PrintMethod. Deze wijzigingen maken extra printerworkflows mogelijk en geven ontwerpers meer controle over het afdrukproces. Hier volgt een overzicht van de wijzigingen:

- Dialoogvensters Pagina-instelling: zowel het standaard als het aangepaste dialoogvenster voor pagina-instelling kan worden weergegeven. De gebruiker kan paginabereiken, papierformaat, afdrukstand en schaal instellen vóór het afdrukken.
- Afdrukweergave: u kunt een weergavemodus maken waarin het papierformaat, de marges en de positie van de inhoud op de pagina precies worden getoond.
- Beperkt afdrukken: ontwerpers kunnen de afdrukopties beperken, zo kunnen ze het aantal afdrukbare pagina's beperken.
- Kwaliteitopties: ontwerpers kunnen de afdrukkwaliteit van een document aanpassen en de gebruiker in staat stellen de resolutie en kleuropties te selecteren.
- Meerdere afdruksessies: u kunt één PrintJob-instantie nu gebruiken voor meerdere afdruksessies. Toepassingen kunnen elke keer dezelfde instellingen weergeven inde dialoogvensters Pagina-instelling en Afdrukken.

Wijzigingen in de afdrukworkflow

De nieuwe workflow voor afdrukken bestaat uit de volgende stappen:

- `new PrintJob()`: hiermee maakt u een PrintJob-instantie (of herbruikt u een bestaande instantie). Vele nieuwe PrintJob-eigenschappen en -methoden, zoals `selectPaperSize()`, zijn beschikbaar voordat de afdruktaak wordt gestart of tijdens het afdrukken.
- `PrintJob.showPageSetupDialog()`: hiermee kunt u het dialoogvenster met pagina-instellingen weergeven zonder een afdruktaak te starten (optioneel).
- `PrintJob.start()` of `PrintJob.start2()`: naast de `start()`-methode wordt de `start2()`-methode gebruikt om het afdrukspoolproces in gang te zetten. Met de `start2()`-methode kunt u aangeven of u het dialoogvenster Afdrukken wilt weergeven. Ook kunt u het weergegeven dialoogvenster aanpassen.
- `PrintJob.addPage()`: hiermee kunt u inhoud toevoegen aan de afdruktaak. Ongewijzigd ten opzichte van bestaand proces.
- `PrintJob.send()` of `PrintJob.terminate()`: de pagina's naar de geselecteerde printer verzenden of de afdruktaak beëindigen zonder deze te verzenden. Afdruktaken worden beëindigd als er een fout optreedt. Als een PrintJob-instantie wordt beëindigd, kan de instantie alsnog opnieuw worden gebruikt. Wanneer u de PrintJob-instantie opnieuw gebruikt, blijven de huidige afdrukinstellingen behouden, ongeacht of de afdruktaak al naar de printer is gestuurd of dat deze is beëindigd.

Het dialoogvenster Pagina-instelling

De `showPageSetupDialog()`-methode geeft het dialoogvenster Pagina-instelling van het besturingssysteem weer als dat door de actieve omgeving wordt ondersteund. Controleer altijd de eigenschap `supportsPageSetupDialog` voordat u deze methode aanroept. Een eenvoudig voorbeeld:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

U kunt de methode desgewenst aanroepen met een PrintUIOptionsklasse-eigenschap waarmee u bepaalt welke opties worden weergegeven in het dialoogvenster Pagina-instelling. U kunt het laagste en hoogste paginanummer instellen. In het volgende voorbeeld worden alleen de eerste drie pagina's afgedrukt:

```
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);
}
```

Afdrukinstellingen wijzigen

U kunt de instellingen van een PrintJob-instantie op elk gewenst moment na het samenstellen van de instantie wijzigen. Zo kunt u bijvoorbeeld de instellingen tussen `addPage()`-aanroepen wijzigen en nadat een afdruktaak is verzonden of beëindigd. Sommige instellingen, zoals de eigenschap `printer` zijn van toepassing op de hele afdruktaak en niet alleen op individuele pagina's. Die instellingen moeten worden ingesteld voordat `start()` of `start2()` wordt aangeroepen.

U kunt de `selectPaperSize()`-methode aanroepen om het standaardpapierformaat in te stellen in de dialoogvensters Pagina-instelling en Afdrukken. U kunt deze methode ook tijdens het uitvoeren van een afdruktaak aanroepen om het papierformaat voor een paginabereik in te stellen. U roept de methode op met gebruik van constante waarden die zijn gedefinieerd in de klasse `PaperSize`. In dit voorbeeld wordt bijvoorbeeld het envelopformaat 10 geselecteerd:

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

Gebruik de eigenschap `printer` om de naam van de printer voor de huidige afdruktaak op te vragen of in te stellen. Dit is standaard de naam van de standaardprinter. De waarde van de eigenschap `printer` is `null` als er geen printers beschikbaar zijn of als het systeem geen ondersteuning biedt voor afdrukken. Als u de printer wilt wijzigen, moet u eerst de lijst met beschikbare printers ophalen met de eigenschap `printers`. Deze eigenschap is een vector waarvan de tekenreekselementen staan voor beschikbare printernamen. Stel de eigenschap `printer` in op een van deze tekenreekswaarden om de desbetreffende printer tot actieve printer te maken. De `printer`-eigenschap van een actieve afdruktaak kan niet worden gewijzigd. Pogingen om deze eigenschap te wijzigen als `start()` of `start2()` met succes is aangeroepen en voordat de afdruktaak is verzonden of voltooid, zullen mislukken. Hier ziet u een voorbeeld van het instellen van deze eigenschap:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

De eigenschap `copies` ontneemt de waarde voor het aantal exemplaren aan de instelling in het dialoogvenster Afdrukken van het besturingssysteem. Met de eigenschappen `firstPage` en `lastPage` vraagt u het paginabereik op. Met de eigenschap `orientation` vraagt u de afdrukstand van het papier op. U kunt deze eigenschappen zo instellen dat ze de waarden in het dialoogvenster Afdrukken overschrijven. In het volgende voorbeeld worden deze eigenschappen ingesteld:

```
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;
```

De volgende instellingen met het kenmerk Alleen-lezen zijn gekoppeld aan `PrintJob` en verschaffen handige informatie over de actieve printerinstellingen:

- `paperArea`: de in punten uitgedrukte rechthoekige afbakening van het printermedium.
- `printableArea`: de in punten uitgedrukte rechthoekige afbakening van het afdrukbare gebied.
- `maxPixelsPerInch`: de fysieke resolutie van de actieve printer, uitgedrukt in pixels per inch.
- `isColor`: of de actieve printer kleurenafdrukken kan maken (het resultaat `true` verschijnt als de actieve printer kleuren kan afdrukken).

Zie “Voorbeeld van afdrukken: Pagina-instelling en afdrukopties” op pagina 990.

Voorbeeld van afdrukken: meerdere pagina's afdrukken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u meer dan één pagina met inhoud afdrukt, kunt u elke pagina met inhoud koppelen aan een andere Sprite (in dit geval `sheet1` en `sheet2`), waarna u voor elke Sprite gebruikmaakt van `PrintJob.addPage()`. In de volgende code wordt deze techniek geïllustreerd:

```
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();
                }
            }
        }
    }
}
```

Voorbeeld van afdrukken: schalen, bijsnijden en reageren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In bepaalde gevallen wilt u misschien bij het afdrukken de grootte (of andere eigenschappen) van een weergaveobject aanpassen om de verschillen tussen de weergave op het scherm en de afdruk op papier te compenseren. Wanneer u de eigenschappen van een weergaveobject aanpast voordat u dit afdrukt (bijvoorbeeld door gebruik te maken van de eigenschappen `scaleX` en `scaleY`), houd er dan rekening mee dat het object wordt bijgesneden als dit een grotere schaal krijgt dan de voor het afdrukgebied gedefinieerde rechthoek. Bovendien zult u de eigenschappen na het afdrukken van de pagina's waarschijnlijk weer op hun oorspronkelijke waarden willen terugzetten.

In de volgende code worden de afmetingen geschaald van het weergaveobject `txt` (maar niet van het groene vak dat als achtergrond fungeert) en wordt het tekstveld bijgesneden op basis van de afmetingen van de gedefinieerde rechthoek. Na het afdrukken wordt het tekstveld teruggezet naar het oorspronkelijke formaat voor weergave op het scherm. Als de gebruiker de afdruktaak annuleert vanuit het dialoogvenster Afdrukken van het besturingssysteem, wordt de inhoud in de Flash-runtime gewijzigd zodat de gebruiker wordt gewaarschuwd dat de afdruktaak is geannuleerd.

```
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;
    }
  }
}
```

Voorbeeld van afdrukken: Pagina-instelling en afdrukopties

Adobe AIR 2 of hoger

In het volgende voorbeeld worden de PrintJob-instellingen voor het aantal exemplaren, het papierformaat (legal) en de paginastand (liggend) geïnitialiseerd. Het dialoogvenster Pagina-instelling wordt eerst weergegeven en daarna wordt de afdruktaak gestart door weergave van het dialoogvenster Afdrukken.

```
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;
        }
    }
}
```

Hoofdstuk 55: Geolocatie

Als een apparaat geolocatie ondersteunt, kunt u deze geolocatie-API gebruiken om de huidige geografische locatie van het apparaat te bepalen. Als het apparaat deze functie ondersteunt, kunt u informatie over de geolocatie ontvangen. Deze informatie omvat onder andere de hoogte, nauwkeurigheid, richting, snelheid en tijdstempel van de laatste wijziging in de locatie.

De Geolocation-klasse verzendt `update`-gebeurtenissen als reactie op de locatiesensor van het apparaat. De `update`-gebeurtenis is een GeolocationEvent-object.

Meer Help-onderwerpen

flash.sensors.Geolocation

flash.events.GeolocationEvent

Wijzigingen in geolocatie bepalen

Als u de geolocatiesensor wilt gebruiken, moet u een Geolocation-object instantiëren en registeren voor de `update`-gebeurtenissen die door het object worden verzonden. De `update`-gebeurtenis is een GeolocationEvent-object. De gebeurtenis heeft acht eigenschappen:

- `altitude`: de hoogte in meters.
- `heading`: de bewegingsrichting (in verhouding tot het werkelijke noorden) in graden.
- `horizontalAccuracy`: de horizontale nauwkeurigheid in meters.
- `latitude`: de latitude in graden.
- `longitude`: de longitude in graden.
- `speed`: de snelheid in meters per seconde.
- `timestamp`: het aantal milliseconden op het moment waarop de gebeurtenis plaatsvindt sinds het runtimeprogramma is geïnitialiseerd.
- `verticalAccuracy`: de verticale nauwkeurigheid in meters.

De eigenschap `timestamp` is een int-object. De overige eigenschappen zijn Number-objecten.

In dit voorbeeld worden de gegevens over de geolocatie in een tekstveld weergegeven:

```
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()
}
```

Als u dit voorbeeld wilt gebruiken, moet u het tekstveld`geoTextField` maken en dit toevoegen aan de weergavelijst voordat u deze code gebruikt.

U kunt het gewenste tijdsinterval voor geolocatiegebeurtenissen aanpassen door de methode `setRequestedUpdateInterval()` van het Geolocation-object aan te roepen. Deze methode neemt één parameter, `interval`, die staat voor het gevraagde update-interval in milliseconden:

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

De werkelijke tijd tussen geolocatie-updates kan groter of kleiner zijn dan deze waarde. Elke wijziging in het update-interval beïnvloedt alle geregistreerde listeners. Als u de methode `setRequestedUpdateInterval()` niet aanroept, ontvangt de toepassing updates op basis van het standaardinterval voor het apparaat.

De gebruiker kan voorkomen dat een toepassing toegang heeft tot gegevens over de geolocatie. Zo kan een iPhone bijvoorbeeld een melding sturen naar de gebruiker wanneer een toepassing probeert toegang te krijgen tot gegevens over de geolocatie. Als reactie hierop kan de gebruiker de toegang tot deze gegevens weigeren. Het Geolocation-object verzendt de gebeurtenis `status` wanneer de gebruiker de toegang tot gegevens over de geolocatie weigert. Bovendien heeft het Geolocation-object de eigenschap `muted`. Deze eigenschap is ingesteld op `true` wanneer de geolocatiesensor niet beschikbaar is. Het Geolocation-object verzendt de gebeurtenis `status` wanneer de waarde van de eigenschap `muted` wordt gewijzigd. De volgende code laat zien hoe u kunt vaststellen dat gegevens over de geolocatie niet beschikbaar zijn.

```
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 = "";
        }
    }
}
```

Opmerking: *Bij iPhones van de eerste generatie (zonder GPS-eenheid) worden slechts af en toe* `update`*-gebeurtenissen verzonden. Bij deze apparaten verzendt een Geolocation-object een of twee initiële* `update`*-gebeurtenissen. Vervolgens worden er alleen* `update`*-gebeurtenissen verzonden wanneer de informatie merkbaar verandert.*

Ondersteuning voor geolocatie

Met de eigenschap `Geolocation.isSupported` kunt u testen of deze functie kan worden gebruikt in de runtimeomgeving:

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

Momenteel wordt de functie voor geolocatie alleen ondersteund in ActionScript-toepassingen voor de iPhone en in Flash Lite 4. Als `Geolocation.isSupported` de waarde `true` heeft tijdens runtime, is er ondersteuning voor de geolocatiefunctie.

Sommige modellen iPhone beschikken niet over een GPS-eenheid. Bij deze modellen worden geolocatiegegevens verkregen via andere methoden (zoals triangulatie via mobiele telefoons). Bij deze modellen, net als bij elke iPhone waarvoor GPS is uitgeschakeld, kan een Geolocation-object niet meer dan een of twee initiële `update`-gebeurtenissen verzenden.

Hoofdstuk 56: Toepassingen geschikt maken voor de internationale markt

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Met het pakket flash.globalization kunt u gemakkelijk internationale software maken die u kunt aanpassen aan de voorschriften van verschillende talen en regio's.

Meer Help-onderwerpen

flash.globalization-pakket

"Toepassingen lokaliseren" op pagina 1015

Charles Bihis: Want to Localize your Flex/AIR Apps?

Grondbeginselen op gebied van toepassingen internationaliseren

De begrippen globalisatie en internationalisatie worden soms door elkaar gebruikt. Maar volgens de meeste definities van deze termen verwijst 'globalisatie' naar een combinatie van zakelijke en technische processen, terwijl 'internationalisatie' alleen naar technische processen verwijst.

Hier volgen de definities van enkele belangrijke begrippen:

Globalisatie Een groot aantal technische en zakelijke processen die nodig zijn om producten en bedrijfsactiviteiten globaal voor te bereiden en te lanceren. Globalisatie bestaat uit technische activiteiten, zoals internationalisatie, lokalisatie en culturalisatie, en uit zakelijke activiteiten, zoals productmanagement, financiële planning, marketing en juridische taken. Globalisatie wordt soms afgekort als *G11n* (dit staat voor de letter G, gevolgd door 11 letters en de letter n, zoals in het Engelse woord Globalisation). *"Globalisatie is een zaak voor bedrijven."*

Internationalisatie Een technisch proces voor het generaliseren van producten, zodat deze meerdere talen, scripts en culturele gebruiken (zoals valuta, sortertingsregels, datumnotaties, numerieke opmaak, enz.) kunnen verwerken zonder het product opnieuw te moeten ontwerpen of te compileren. Het proces kan worden opgesplitst in twee soorten activiteiten: facilitatie en lokalisatie. Internationalisatie wordt in het Engels ook wel *world-readiness* genoemd en wordt in deze taal afgekort tot *I18n*. *"Internationalisatie is een zaak van engineers."*

Lokalisatie Het aanpassen van een product of service voor een bepaalde taal en cultuur en voor een gewenst lokaal profiel. Lokalisatie wordt in het Engels wel afgekort tot *L10n*. *"Lokalisatie is een zaak van vertalers."*

Aanpassen aan cultuur Een technisch proces voor het ontwikkelen of aanpassen van specifieke functies voor de unieke behoeften van een cultuur. Denk hierbij bijvoorbeeld aan de publicatiefuncties voor Japan die beschikbaar zijn in Adobe InDesign en aan de ondersteuning voor de Hanko-functie in Adobe Acrobat.

Hier volgt de definitie van elke andere belangrijke begrippen die te maken hebben met internationalisatie:

Tekenset De tekens die door een taal of door een groep talen worden gebruikt. Een tekenset bestaat uit nationale tekens, speciale tekens (zoals interpunctie en wiskundige symbolen), numerieke cijfers en tekens voor computerbesturing.

Sorteren Het sorteren van tekst in de juiste volgorde voor een bepaalde landinstelling.

Landinstelling Een waarde die staat voor de taalkundige en culturele gebruiken van een geografisch, politiek of cultureel gebied (vaak gaat het hierbij om een land). Deze waarde wordt aangegeven door een unieke landinstellings-id. Met de landinstellings-id kunt u zoeken naar een gegevensset die ondersteuning biedt voor de gewenste landinstellingen. Deze ondersteuning is van toepassing op maateenheden en op de parsering en notatie van getallen en datums, enzovoort.

Resourcepakket Een opgeslagen set met specifiek voor de landinstelling geldende elementen. Deze set wordt gemaakt voor de regio waarin de toepassing wordt gebruikt. Een resourcepakket bevat over het algemeen alle tekstelementen van de gebruikersinterface van de toepassing. Deze elementen worden in het pakket vertaald in de gewenste taal voor de opgegeven landinstelling. Dit pakket kan ook andere instellingen bevatten die de lay-out of de functionaliteit van de gebruikersinterface voor een specifieke landinstelling veranderen. Het resourcepakket kan ook andere mediatypen (of verwijzingen naar mediatypen) bevatten die specifiek zijn voor de opgegeven landinstelling.

Overzicht van het flash.globalization-pakket

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Het pakket flash.globalization benut de mogelijkheden voor culturele ondersteuning van het onderliggende besturingssysteem. U kunt zo gemakkelijker toepassingen wegschrijven waarin de culturele gebruiken van afzonderlijke gebruikers worden nageleefd.

Hier volgt een overzicht van de hoofdklassen in het pakket:

- De klasse Collator bepaalt het sorteren en afstemmen van tekenreeksen
- De klasse CurrencyFormatter zet getallen om in valuta's en verwerkt bedragen in valuta en symbolen op basis van invoerreeksen
- De klasse DateTimeFormatter zet datumwaarden om in de juiste notatie
- De klasse LocaleID haalt informatie op over een bepaalde landinstelling
- De klasse NumberFormatter verwerkt numerieke waarden en zet deze om in de juiste notatie
- De klasse StringTools verwerkt de hoofdlettergevoelige omzetting van tekenreeksen afhankelijk van de landinstelling

Het flash.globalization-pakket en lokalisatie van resources

De resourcelokalisatie vormt geen onderdeel van het flash.globalization-pakket. U kunt wel de landinstellings-id van flash.globalization gebruiken als hoofdwaarde voor het ophalen van gelokaliseerde resources met behulp van andere technieken. Zo kunt u de toepassingsresources die zijn gemaakt met Flex lokaliseren met de klassen ResourceManager en ResourceBundle. Zie Flex-toepassingen lokaliseren voor meer informatie.

Adobe AIR 1.1 bevat enkelen functies voor het lokaliseren van AIR-toepassingen, zoals besproken in “AIR-toepassingen lokaliseren” op pagina 1016.

Een algemene aanpak voor internationalisatie van een toepassing

De volgende procedure beschrijft een aanpak op hoog niveau voor de internationalisering van een toepassing met het flash.globalization-pakket:

1 Bepaal de landinstellingen en stel deze in.

2 Maak een instantie van een serviceklasse (Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter of StringTools).

3 Controleer op fouten en fallbacks met eigenschappen van lastOperationStatus.

4 Pas een opmaak toe en geef de informatie weer met specifieke instellingen en waarden voor het desbetreffende land.

De volgende stap bestaat uit het laden en weergeven van tekenreeksen en UI-resources die specifiek zijn voor het land. Deze stap bevat taken zoals:

- De afmetingen van de UI met de functie voor automatische lay-out aanpassen aan de lengte van tekenreeksen.
- De juiste lettertypen kiezen en ondersteuning bieden voor fallback-lettertypen.
- Andere schrijfsystemen ondersteunen met de FTE-tekstengine.
- Ervoor zorgen dat editors voor invoermethoden correct worden verwerkt.

Controleren op fouten en fallbacks

De flash.globalization-serviceklassen volgen een vergelijkbaar patroon voor foutidentificatie. De klassen delen ook een fallbackpatroon, waarbij wordt omgeschakeld van een niet-beschikbare landinstelling naar een landinstelling die wordt ondersteund door het besturingssysteem van de gebruiker.

In het volgende voorbeeld wordt getoond hoe u controleert op fouten en fallbacks bij het instantiëren van serviceklassen. Elke service-klasse heeft een lastOperationStatus-eigenschap die aangeeft of de weergegeven fouten of waarschuwingen worden veroorzaakt door de meest recente methodeaanroep.

```
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);
    }
}
```

In dit voorbeeld wordt een bericht getraceerd als een andere landinstellings-id wordt gebruikt als fallback of als er een fout is opgetreden. Indien nodig kan uw toepassing extra logica voor foutafhandeling uitvoeren. Zo kunt u bijvoorbeeld een bericht weergeven aan de gebruiker of de toepassing forceren om een bepaalde, ondersteunde landinstelling te gebruiken.

De landinstelling bepalen

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Met een landinstelling wordt een specifieke combinatie van taal en culturele conventies voor een land of regio aangeduid.

De landinstellings-id kan probleemloos worden beheerd als tekenreeks. Met de LocaleID-klasse kunt u extra informatie ophalen die te maken heeft met een landinstelling.

U kunt een LocaleID als volgt maken:

```
var locale:LocaleID = new LocaleID("es-MX");
```

Nadat het LocaleID-object is gemaakt, kunt u de gegevens over de landinstellings-id ophalen. Gebruik de methoden `getKeysAndValues()`, `getLanguage()`, `getRegion()`, `getScript()`, `getVariant()` en `isRightToLeft()`, en de eigenschap `name`.

De waarden die u ophaalt met deze methoden en eigenschappen bevatten extra informatie over de landinstellingen die niet rechtstreeks kan worden opgehaald van de landinstellings-id.

Wanneer een toepassing een service maakt die rekening houdt met de landinstelling, zoals bijvoorbeeld een functie voor datumnotatie, moet de toepassing aangeven welke landinstellingen worden gebruikt. De lijst met ondersteunde landinstellingen verschilt per besturingssysteem. De aangevraagde landinstellingen zijn daarom niet altijd beschikbaar.

Flash Player zoekt eerst naar overeenkomsten op basis van de taalcode van de landinstelling die u wilt instellen. Vervolgens probeert het programma om de landinstelling te verfijnen door te zoeken naar een identiek schrijfsysteem (schrift) en regio. Bijvoorbeeld:

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

In dit voorbeeld heeft de `LocaleID()`-constructor gegevens over de landinstelling opgehaald die het meest overeenkomen met de taalcode “es” voor de desbetreffende gebruiker.

De landinstellings-id instellen

U kunt de huidige landinstellingen van een toepassing op een aantal verschillende manieren instellen, waaronder de volgende:

- Leg de code voor een enkele landinstellings-id vast in de toepassing. Deze aanpak komt vaak voor, maar biedt geen ondersteuning voor de internationalisatie van de toepassing.
- Gebruik de voorkeuren van de landinstellings-id van het besturingssysteem van de gebruiker of browser, of gebruik andere gebruikersvoorkeuren. Deze werkwijze leidt gewoonlijk tot de beste landinstellingen voor de gebruiker, maar is niet altijd nauwkeurig. Er bestaat een risico dat de instellingen van het besturingssysteem niet overeenkomen met de werkelijke voorkeuren van de gebruiker. De gebruiker kan bijvoorbeeld een computer met iemand delen, waardoor hij de voorkeurslandinstellingen niet zelf kan aanpassen.
- Nadat u de landinstellings-id hebt gebaseerd op de gebruikersvoorkeuren laat u de gebruiker kiezen uit een lijst met ondersteunde landinstellingen. Deze strategie is vaak de beste optie als uw toepassing meerdere landinstellingen ondersteunt.

U kunt deze derde optie als volgt implementeren:

1. Haal een lijst op met de voorkeursinstellingen van de gebruiker voor landen of talen via een gebruikersprofiel, via de instellingen van de browser of van het besturingssysteem of via een cookie. (Uw toepassing moet deze logica zelf implementeren. De flash.globalization-bibliotheek biedt geen ondersteuning voor het rechtstreeks inlezen van deze voorkeuren.)

2 Bepaal welke landinstellingen worden ondersteund door uw toepassing en stel de beste optie in als standaardwaarde. Met de methode LocaleID.determinePreferredLocales() vindt u de beste landinstellingen voor een gebruiker, op basis van de voorkeurslandinstellingen en de landinstellingen die worden ondersteund door het besturingssysteem.

3 Biedt de gebruiker de mogelijkheid om de standaardlandinstelling te wijzigen als deze niet overeenkomt met de wensen van de gebruiker.

Beperkingen van andere klassen voor landinstelling en taal

Met de klasse `fl.lang.Locale` kunt u teksttekenreeksen op basis van een landinstelling vervangen. Hiervoor gebruikt u resourcepakketten met tekenreekswaarden. Deze klasse biedt echter geen ondersteuning voor de internationalisatie van functies zoals getal-, valuta- of datumnotatie, sorteren, zoekovereenkomsten en dergelijke. Bovendien is deze klasse alleen beschikbaar in Flash Professional.

U kunt de huidige taalcode-instelling van het besturingssysteem ook ophalen met de eigenschap `flash.system.Capabilities.language`. Deze eigenschap haalt echter alleen de ISO 639-1-taalcode op die uit twee letters bestaat en niet de volledige landinstellings-id. Bovendien wordt alleen een specifieke set met landinstellingen ondersteund.

Met AIR 1.5 kunt u de eigenschap `flash.system.Capabilities.languages` gebruiken. Hiermee beschikt u over een array met talen voor de gebruikersinterface waaraan de gebruiker zijn voorkeur geeft. Deze eigenschap heeft daarom niet de beperkingen van `Capabilities.language`.

Getallen opmaken

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

De weergave van numerieke waarden varieert enorm van regio tot regio. Hier ziet u de notatie van het getal 123456,78 voor verschillende landinstellingen:

Landinstelling	Getalnotatie
en-US (Engels, Verenigde Staten)	-123,456.78
de-DE (Duits, Duitsland)	-123.456,78
fr-FR (Frans, Frankrijk)	-123 456,78
de-CH (Duits, Zwitserland)	-123'456.78
en-IN (Engels, India)	-1,23,456.78
Veel Arabische landinstellingen	123,456.78-

Veel factoren zijn van invloed op de getalnotatie, waaronder:

- Scheidingstekens. Het decimale scheidingsteken staat tussen het gedeelte van het getal dat bestaat uit gehele cijfers en het breukgedeelte van het getal. Dit kan een punt, komma of ander teken zijn. Het groeperingsscheidingteken of scheidingsteken voor duizendtallen kan een punt, komma, vaste spatie of ander teken zijn.
- Groeperingspatronen. Het aantal cijfers tussen elk groeperingsscheidingsteken links van de decimale komma kan twee, drie of een ander getal zijn.

- Indicator voor negatieve getallen. Negatieve getallen kunnen worden aangeduid met een minteken links of rechts van het getal, of het getal kan tussen haakjes worden geplaatst bij financiële toepassingen. Het negatieve getal 19 komt voor als -19, 19- of (19).
- Voorloopnullen en volgnullen. In sommige culturele conventies worden voorloopnullen of volgnullen toegevoegd aan de getalnotatie. Zo kan de waarde 0,17 onder andere worden weergegeven als ,17 of als 0,17 of 0,170.
- Cijfersets. In veel talen, waaronder het Hindi, Arabisch en Japans worden verschillende cijfersets gebruikt. Het flash.globalization-pakket biedt ondersteuning voor alle cijfersets die kunnen worden toegewezen aan de cijfers 0-9.

Bij de notatie van numerieke waarden wordt met al deze factoren rekening gehouden door de klasse NumberFormatter.

De klasse NumberFormatter gebruiken

Met de NumberFormatter-klasse bepaalt u de notatie van numerieke waarden (van het type int, uint of Number) volgens de conventies van een bepaalde landinstelling.

Het volgende voorbeeld beschrijft de eenvoudigste manier om een getalnotatie toe te passen op basis van de standaardopmaakkenmerken van het besturingssysteem van de gebruiker:

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

Het resultaat is afhankelijk van de landinstellingen van de gebruiker en van de gebruikersvoorkeuren. Als de gebruiker de landinstelling fr-FR heeft, krijgt het getal de volgende notatie:

-123.456,789

Als u de getalnotatie alleen wilt baseren op een specifieke landinstelling, onafhankelijk van de gebruikersinstellingen, moet u de naam van de landinstelling specifiek instellen. Bijvoorbeeld:

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

In dit geval is het resultaat:

-123'456.789

De methode formatNumber() neemt een Number als parameter. De klasse NumberFormatter heeft ook een formatInt()-methode die een int neemt als invoer, en een formatUint()-methode die een uint neemt als invoer.

U kunt de logica voor de opmaak expliciet beheren door de eigenschappen van de klasse NumberFormatter in te stellen, zoals in dit voorbeeld:

```
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)
```

In dit voorbeeld wordt eerst een NumberFormatter-object gemaakt, waarna de volgende stappen worden uitgevoerd:

- als notatie voor negatieve getallen worden haakjes toegepast (zoals bij financiële toepassingen);
- het aantal cijfers na het decimale scheidingsteken wordt ingesteld op 5;
- volgnullen worden gebruikt om vijf decimale cijfers te garanderen;
- het decimale scheidingsteken wordt ingesteld op een komma;

- de notatiefunctie mag geen groeperingsscheidingstekens toepassen.

Opmerking: *Wanneer sommige van deze eigenschappen veranderen, komt de resulterende getalnotatie niet meer overeen met de voorkeursnotatie voor de opgegeven landinstelling. Gebruik sommige van deze eigenschappen alleen wanneer de landinstellingen niet belangrijk zijn, bijvoorbeeld wanneer een enkel aspect van de notatie in detail moet worden beheerd, zoals het aantal volgnullen, of wanneer de gebruiker de wijziging direct heeft aangevraagd, bijvoorbeeld via het Windows Configuratiescherm.*

Reeksen met numerieke waarden verwerken

Met de NumberFormatter-klasse kunt u ook numerieke waarden ophalen van tekenreeksen die voldoen aan de vereisten voor landspecifieke notaties. Met de methode NumberFormatter.parseNumber() haalt u een enkele numerieke waarde op van een tekenreeks. Bijvoorbeeld:

```
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
```

Met de methode parseNumber() kunt u tekenreeksen afhandelen die alleen cijfers en getalnotatietekens bevatten, zoals het minteken en scheidingstekens. Als de tekenreeks andere tekens bevat, wordt een foutcode ingesteld:

```
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
```

Als u getallen wilt extraheren uit tekenreeksen met extra alfabetische tekens, gebruikt u de NumberFormatter.parse()-methode:

```
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
```

De methode parse() retourneert een NumberParseResult-object met de geparseerde numerieke waarde in de value-eigenschap. De startIndex-eigenschap geeft de index aan van het eerste numerieke teken dat wordt aangetroffen. Met de eigenschappen startIndex en endIndex kunt u de gedeelten van de tekenreeks extraheren die voor en na de cijfers liggen.

Valutawaarden opmaken

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

De notaties voor valutawaarden variëren, net als getalnotaties. Hier ziet u de notatie voor het bedrag $123456,78 voor verschillende landinstellingen:

Landinstelling	Getalnotatie
en-US (Engels, Verenigde Staten)	$123,456.78
de-DE (Duits, Duitsland)	123.456,78 $
en-IN (Engels, India)	$ 1,23,456.78

Bij de notatie van valuta's zijn dezelfde factoren betrokken als bij de getalnotatie, plus de volgende factoren:

- ISO-code voor valuta's. De ISO 4217-valutacode bestaat uit drie letters en geeft de landinstelling aan die wordt gebruikt, zoals USD of EUR.
- Valutasymbool. Het valutasymbool of de tekenreeks voor de werkelijk gebruikte landinstelling, zoals € of $.
- Notatie voor negatieve valutabedragen. Hiermee wordt de locatie bepaald van het valutasymbool en het negatieve symbool of haakjes in verband met het numerieke deel van het valutabedrag.
- Notatie voor positieve valutabedragen. Hiermee wordt de locatie bepaald van het valutasymbool ten opzichte van het numerieke onderdeel van het valutabedrag.

De klasse CurrencyFormatter gebruiken

Met de CurrencyFormatter-klasse kunt u numerieke waarden noteren als tekenreeksen waarin valutatekenreeksen en opgemaakte getallen zijn opgenomen, conform de conventies van een bepaalde landinstelling.

Wanneer u een nieuw CurrencyFormatter-object instantieert, wordt de standaardvaluta voor de desbetreffende landinstelling ingesteld.

In het volgende voorbeeld ziet u dat bij een CurrencyFormatter-object dat met een Duitse landinstelling is gemaakt, wordt aangenomen dat het valutabedrag in euro staat:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

In de meeste gevallen moet u niet uitgaan van de standaardvaluta voor een landinstelling. Als de standaardlandinstelling van de gebruiker niet wordt ondersteund, wordt door de CurrencyFormatter-klasse een andere landinstelling als fallback toegewezen. Bij deze fallback kan een andere standaardvaluta van toepassing zijn. Bovendien wilt u waarschijnlijk ook dat de valutanotaties correct zijn in de ogen van de gebruiker, zelfs als de bedragen niet in de lokale valuta van de gebruiker staan. Zo wil een Canadese gebruiker bijvoorbeeld dat bedragen voor een Duits bedrijf in euro worden weergegeven, maar worden opgemaakt volgens de Canadese stijl.

De methode CurrencyFormatter.setCurrency() geeft de exacte valutatekenreeks en -symbool die moeten worden gebruikt.

In het volgende voorbeeld worden valutabedragen in euro weergegeven voor gebruikers in het Franse gedeelte van Canada:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Met de methode setCurrency() voorkomt u verwarring doordat u niet-ambigue valutasymbolen toepast. Bijvoorbeeld:

```
cf.setCurrency("USD","US$");
```

De methode format() geeft standaard een ISO 4217-valutacode weer die uit drie tekens bestaat, in plaats van het valutasymbool. ISO 4217-codes zijn niet ambigu en vereisen geen lokalisatie. Veel gebruikers geven echter de voorkeur aan valutasymbolen en niet aan de ISO-codes.

Met de CurrencyFormatter-klasse kunt u beter bepalen welk symbool moet worden gebruikt in een opgemaakte valutatekenreeks: een valutasymbool, zoals het dollar- of eurosymbool, of een ISO-valutatekenreeks die uit drie tekens bestaat, zoals USD of EUR. Een bedrag in Canadese dollars kan bijvoorbeeld worden weergegeven als $200 voor een gebruiker in Canada. Voor een gebruiker in de Verenigde Staten kan dit bedrag worden weergegeven als CAD 200. Met de methode formattingWithCurrencySymbolIsSafe() kunt u nagaan of het valutasymbool van het bedrag ambigu of onjuist is, gezien de landinstellingen van de gebruiker.

In het volgende voorbeeld wordt een waarde in euro opgemaakt in een notatie voor de landinstelling en-US. Afhankelijk van de landinstelling van de gebruiker, wordt bij de uitvoertekenreeks ofwel de ISO-valutacode of het valutasymbool toegepast.

```
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
}
```

Reeksen met valutawaarden verwerken

Met de CurrencyFormatter-klasse kunt u ook een valutabedrag en -tekenreeks ophalen van een invoertekenreeks die voldoet aan de specifieke vereisten voor de notatie van de landinstelling. De methode CurrencyFormatter.parse() slaat het geparseerde bedrag en de valutatekenreeks op in een CurrencyParseResult-object, zoals hier wordt aangegeven:

```
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
```

Het valutatekenreeksgedeelte van de invoertekenreeks kan een valutasymbool, een ISO-valutacode en extra teksttekens bevatten. De posities van de valutatekenreeks, de indicator voor een negatief bedrag en de numerieke waarde komen overeen met de notaties die zijn opgegeven door de eigenschappen negativeCurrencyFormat en positiveCurrencyFormat. Bijvoorbeeld:

```
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 $
```

In dit voorbeeld bestaat de invoertekenreeks uit een valutatekenreeks, gevolgd door een minteken en een bedrag. Volgens de standaardwaarde voor negativeCurrencyFormat voor de landinstelling en-US moet de negatieve indicator echter vooraan staan. Als gevolg hiervan genereert de parse()-methode een fout en is de geparseerde waarde NaN.

Nadat de waarde van negativeCurrencyFormat is ingesteld op 2, waarmee wordt aangegeven dat de valutatekenreeks vooraan staat, kan de parse()-methode succesvol worden uitgevoerd.

De datum en tijd opmaken

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

De datum- en tijdnotaties verschillen ook aanzienlijk per regio. Hieronder ziet u hoe de tweede dag van januari 1962 (met als tijdstip 13:01) wordt weergegeven in de korte notatie van een aantal landinstellingen:

Landinstelling	Datum- en tijdnotatie
en-US (Engels, Verenigde Staten)	1/2/62 1:01pm
fr-FR (Frans, Frankrijk)	2/1/62 13:01
ja-JP (Japans, Japan)	1962/2/1 13:01

De klasse DateTimeFormatter gebruiken

Met de klasse DateTimeFormatter kunt u Date-waarden weergeven in datum- en tijdtekenreeksen volgens de conventies van een bepaalde landinstelling.

De notatie wordt bepaald door een patroontekenreeks met daarin een letterreeks die wordt vervangen met datum- en tijdwaarden. In het patroon "yyyy/MM" worden de letters "yyyy" bijvoorbeeld vervangen door een viercijferig jaar, gevolgd door het teken "/" en door een tweecijferige maand.

U kunt de patroontekenreeks precies instellen met de methode setDateTimePattern(). Het wordt echter aanbevolen dat het patroon automatisch wordt ingesteld op basis van de landinstellingen van de gebruiker en de voorkeuren van het besturingssysteem. Hierdoor is het resultaat geschikt voor de desbetreffende cultuur.

DateTimeFormatter kan de datum en tijd weergeven in drie standaardstijlen (LONG, MEDIUM and SHORT). Verder kan het patroon CUSTOM worden gebruikt. U kunt één bepaalde stijl gebruiken voor de datum en een andere voor de tijd. De werkelijke patronen die voor elke stijl worden gebruikt variëren in kleine mate per besturingssysteem.

U kunt de stijlen opgeven wanneer u een DateTimeFormatter-object maakt. Als de stijlparameters niet worden opgegeven, worden ze standaard ingesteld op DateTimeStyle.LONG. U kunt de stijlen later wijzigen met de methode setDateTimeStyles(), zoals in het volgende voorbeeld wordt aangegeven:

```
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
```

Namen voor dagen en maanden lokaliseren

In veel toepassingen worden lijsten met de namen van maanden en weekdagen gebruikt in agendaweergaven of keuzelijsten.

Met de methode DateTimeFormatter.getMonthNames() kunt u een gelokaliseerde lijst met de namen voor de maanden ophalen. Afhankelijk van het besturingssysteem zijn volledige namen en afkortingen beschikbaar. Geef de waarde DateTimeNameStyle.FULL door om de volledige namen voor maanden op te halen. Met de waarden DateTimeNameStyle.LONG_ABBREVIATION of DateTimeNameStyle.SHORT_ABBREVIATION krijgt u korte versies.

In sommige talen verandert de naam van de maand (in de genitief) wanneer deze naast de waarde voor de dag wordt geplaatst in een datumnotatie. Als u de naam van de maand zelfstandig wilt gebruiken, moet u de waarde DateTimeNameContext.STANDALONE doorgeven aan de methode getMonthNames(). Als u de namen van de maand wilt gebruiken in datumnotaties, moet u de waarde DateTimeNameContext.FORMAT doorgeven.

```
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.
```

Met de methode DateTimeFormatter.getWeekdayNames() beschikt u over een gelokaliseerde lijst met de namen voor de weekdagen. De methode getWeekdayNames() accepteert dezelfde nameStyle- en context -parameters als de methode 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.
```

Bovendien retourneert de methode getFirstWeekday() de indexwaarde van de eerste weekdag die traditioneel wordt toegepast in de geselecteerde landinstellingen.

Reeksen sorteren en vergelijken

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Sorteren is het proces waarbij items in de juiste volgorde worden geplaatst. De regels voor het sorteren verschillen per landinstelling. Er gelden weer andere regels als u lijsten of overeenkomstige items sorteert, zoals bij een zoekalgoritme voor tekst.

Bij het sorteren zijn kleine verschillen, zoals hoofdletters of kleine letters en het gebruik van diakritische tekens zoals accenten vaak heel belangrijk. De letter ö (o met een trema) wordt in het Frans en Engels beschouwd als praktisch gelijk aan de gewone letter o. In het Zweeds volgt deze letter echter altijd op de letter z. In het Frans (en ook in sommige andere talen) bepaalt het accent ook de plaats van een woord in een gesorteerde lijst.

Bij het zoeken wilt u vaak de verschillen in hoofdletters en kleine letters of het gebruik van diakritische tekens negeren. Zo hebt u namelijk een grotere kans om relevante overeenkomsten te vinden. Als u bijvoorbeeld zoekt naar de tekens “cote” in een Frans document, worden in dat geval ook de woorden “cote”, “côte” en “coté” geretourneerd.

De klasse Collator gebruiken

De belangrijkste methoden van de Collator-klasse zijn de compare()-methode, voornamelijk gebruikt voor sorteren, en de equals()-methode, gebruikt om te zoeken naar overeenkomstige waarden.

In het volgende voorbeeld wordt het verschil tussen de compare()-methode en equals()-methode duidelijk.

```
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
}
```

In het voorbeeld wordt eerste een Collator-object gemaakt in de modus SORTING voor de landinstelling Frans-Frankrijk. Vervolgens worden twee woorden gesorteerd die alleen verschillende diakritische tekens gebruiken. Dit geeft aan dat de SORTING-vergelijking een onderscheid maakt tussen tekens met of zonder accent.

Het sorteren gebeurt doordat een referentie naar de sort()-methode van het Collator-object wordt doorgegeven naar de Array.sort()-methode. Deze techniek biedt een van de meest efficiënte manieren om een Collator-object toe te passen bij het bepalen van de sorteervolgorde.

In het voorbeeld wordt nu een Collator-object gemaakt in de modus MATCHING. Wanneer beide woorden worden vergeleken door het Collator-object, worden ze als gelijkwaardig behandeld. Dit toont aan dat de MATCHING-vergelijking tekens met en zonder accent als identiek beschouwt.

Het gedrag van de Collator-klasse aanpassen

Standaard gebruikt de Collator-klasse vergelijkingsregels voor tekenreeksen die worden verkregen van het besturingssysteem en die zijn gebaseerd op de landinstellingen en de systeemvoorkeuren van de gebruiker. U kunt het gedrag van de methoden compare() en equals() aanpassen door expliciet bepaalde eigenschappen in te stellen. De volgende tabel geeft een overzicht van de eigenschappen en het effect ervan op vergelijkingen:

Collator-eigenschap	**Effect**
numericComparison	Bepaalt of digitale tekens worden behandeld als getallen of als tekst.
ignoreCase	Bepaalt of verschillen tussen hoofdletters en kleine letters worden genegeerd.
ignoreCharacterWidth	Bepaalt of vormen met volledige breedte of met halve breedte van sommige Chinese en Japanse tekens worden beschouwd als gelijkwaardig.
ignoreDiacritics	Bepaalt of tekenreeksen die dezelfde basistekens gebruiken maar verschillend zijn met betrekking tot accenten of andere diakritische tekens worden beschouwd als gelijkwaardig.
ignoreKanaType	Bepaalt of tekenreeksen die alleen verschillen in het type Kana-teken dat wordt gebruikt, worden beschouwd als gelijkwaardig.
ignoreSymbols	Bepaalt of symbolen, zoals spaties, valutasymbolen, wiskundige symbolen en andere symbolen worden genegeerd.

De volgende code laat zien dat het instellen van de eigenschap ignoreDiacritics op de waarde True, van invloed is op de sorteervolgorde van een lijst met Franse woorden:

```
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
```

Omzetten in hoofdletters/kleine letters

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

Talen verschillen ook in de regels voor het omzetten van hoofdletters naar kleine letters en omgekeerd.

Bij de meeste talen die het Latijnse alfabet gebruiken is de kleine letter van de hoofdletter “I” gelijk aan “i”. Bij sommige talen (zoals Turks en Azeri) bestaat er de extra letter “ı”, zonder punt. Als gevolg hiervan wordt de kleine puntloze letter “ı” omgezet in de hoofdletter ”I”. De kleine letter “i” (met punt) wordt omgezet in de hoofdletter “İ” (met punt).

De StringTools-klasse bevat methoden waarmee dergelijke omzettingen worden uitgevoerd op basis van taalspecifieke regels.

De klasse StringTools gebruiken

De StringTools-klasse biedt twee methoden voor de omzetting van hoofdletters naar kleine letters en omgekeerd: toLowerCase() en toUpperCase(). U maakt een StringTools-object door de constructor aan te roepen met een landinstellings-id. De StringTools-klasse haalt de omzettingsregels voor de desbetreffende landinstelling (of een landinstelling die als fallback dient) op van het besturingssysteem. Het omzettingsalgoritme kan niet verder worden aangepast.

In het volgende voorbeeld worden de methoden toUpperCase() en toLowerCase() gebruikt bij het omzetten van een Duitse frase met de letter “ß”.

```
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
```

Met de methode toUpperCase() wordt de kleine letter “ß” omgezet in de hoofdletters “SS”. Deze omzetting geldt echter in één richting. Wanneer de letters “SS” worden omgezet naar kleine letters, is het resultaat “ss” en niet “ß”.

Voorbeeld: een toepassing voor het volgen van aandelenkoersen internationaliseren

Flash Player 10.1 of hoger, Adobe AIR 2.0 of hoger

De toepassing Global Stock Ticker haalt fictieve gegevens over aandelen op in drie verschillende aandelenmarkten: de Verenigde Staten, Japan en Europa. Vervolgens worden de gegevens weergegeven. De gegevens krijgen een notatie die overeenkomt met de conventies van de verschillende landinstellingen.

In dit voorbeeld wordt aandacht besteed aan de volgende functies van het flash.globalization-pakket:

- Getalnotatie op basis van landinstellingen
- Valutanotatie op basis van landinstellingen
- Instellen van ISO-valutacode en valutasymbolen
- Datumnotatie op basis van landinstellingen
- Ophalen en weergeven van correcte afkortingen voor de namen van de maanden

Zie www.adobe.com/go/learn_programmingAS3samples_flash_nl als u de toepassingsbestanden voor dit voorbeeld wilt downloaden. De toepassingsbestanden van Global Stock Ticker staan in de map Samples/GlobalStockTicker. De toepassing bestaat uit de volgende bestanden:

Bestand	Beschrijving
GlobalStockTicker.mxml of GlobalStockTicker.fla	De gebruikersinterface voor de toepassing voor Flex (MXML) of Flash (FLA).
styles.css	Stijlen voor de gebruikersinterface van de toepassing (alleen Flex).
com/example/programmingas3/stockticker/flex/FinGraph.mxml	Een MXML-component waarmee een grafiek met gesimuleerde aandelengegevens wordt weergegeven, alleen voor Flex.
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	Een Document-klasse met de logica voor de gebruikersinterface voor de toepassing (alleen Flash).
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Een weergavefunctie voor aangepaste cellen voor de Flash DataGrid-component (alleen Flash).
com/example/programmingas3/stockticker/FinancialGraph.as	Een ActionScript-klasse waarmee een grafiek met gesimuleerde aandelengegevens kan worden getekend.
com/example/programmingas3/stockticker/Localizer.as	Een ActionScript-klasse waarmee de landinstellingen en de valuta worden beheerd en waarmee de gelokaliseerde getal-, valuta- en datumnotaties worden afgehandeld.
com/example/programmingas3/stockticker/StockDataModel.as	Een ActionScript-klasse waarin alle gegevens van het Global Stock Ticker-voorbeeld zijn opgeslagen.

De gebruikersinterface en voorbeeldgegevens

De hoofdelementen van de gebruikersinterface zijn:

- een keuzelijst met invoervak om een landinstelling te selecteren
- een keuzelijst met invoervak om een markt te selecteren
- een gegevensraster waarin de gegevens voor zes bedrijven in elke markt staan
- een grafiek met gesimuleerde historische gegevens voor de aandelen van het geselecteerde bedrijf

Alle voorbeeldgegevens over landinstellingen, markten en bedrijfsaandelen zijn door de toepassing opgeslagen in de StockDataModel-klasse. Een werkelijke toepassing zou de gegevens ophalen van een server en deze vervolgens opslaan in een klasse als StockDataModel. In dit voorbeeld zijn alle gegevens vastgelegd in de StockDataModel-klasse.

Opmerking: *De gegevens die worden weergegeven in de financiële grafiek hoeven niet overeen te komen met de gegevens die in het DataGrid-besturingselement worden weergegeven. De grafiek wordt telkens opnieuw getekend wanneer een ander bedrijf wordt geselecteerd. De grafiek dient alleen ter illustratie.*

De landinstelling opgeven

Na enkele initiële installatiewerkzaamheden, roept de toepassing de methode Localizer.setLocale() aan om opmaakobjecten voor de standaardlandinstelling te maken. De methode setLocale() wordt ook telkens geroepen wanneer de gebruiker een nieuwe waarde selecteert van de keuzelijst met landinstellingen.

```
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);
    }
}
```

Eerst maakt de setLocale()-methode een LocaleID-object. Door dit object kunnen details over de werkelijke landinstelling later gemakkelijker worden opgehaald, indien nodig.

Vervolgens worden nieuwe NumberFormatter-, CurrencyFormatter- en DateTimeFormatter-objecten gemaakt voor de landinstelling. Telkens nadat een Formatter-object wordt gemaakt, wordt de methode traceError() aangeroepen. Deze methode geeft foutberichten en waarschuwingen weer in de console als er een probleem optreedt met de aangevraagde landinstelling. (Een werkelijke toepassing moet reageren op basis van deze fouten en niet alleen maar de fouten traceren.)

Nadat het CurrencyFormatter-object is gemaakt, worden de ISO-valutacode van de opmaakfunctie, het valutasymbool en de fractionalDigits-eigenschappen ingesteld op eerdere vastgestelde waarden door de methode setLocale(). (Deze waarde worden telkens ingesteld wanneer de gebruiker een nieuwe markt kiest uit de keuzelijst met markten.)

Nadat het DateTimeFormatter-object is gemaakt, haalt de methode setLocale() ook een array met afkortingen van de gelokaliseerde namen voor de maanden op.

De gegevens opmaken

De opgemaakte aandelengegevens worden aangeleverd in een DataGrid-besturingselement. De individuele DataGrid-kolommen roepen een labelfunctie aan waarmee de kolomwaarde wordt opgemaakt met het juiste opmaakobject.

In de Flash-versie worden de DataGrid-kolommen bijvoorbeeld ingesteld met de volgende code:

```
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;
```

De Flex-versie in het voorbeeld declareert het DataGrid in MXML. Deze versie definieert ook vergelijkbare labelfuncties voor elke kolom.

De labelFunction-eigenschappen verwijzen naar de volgende functies, die de opmaakmethoden van de Localizer-klasse aanroepen:

```
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);
}
```

Vervolgens worden de juiste opmaakelementen geconfigureerd en opgehaald door de Localizer-methoden:

```
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);
}
|
```

Hoofdstuk 57: Toepassingen lokaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Lokalisatie is het proces waarbij assets worden toegevoegd voor de ondersteuning van meerdere landinstellingen. Een landinstelling is de combinatie van een taal en een landcode. Zo verwijst en_US bijvoorbeeld naar de Engelse taal zoals die wordt gesproken in de Verenigde Staten, en fr_FR naar de Franse taal zoals die wordt gesproken in Frankrijk. Als u een toepassing wilt lokaliseren voor deze landinstellingen, moet u beschikken over twee sets assets: een voor de landinstelling en_US en een voor de landinstelling fr_FR.

Verschillende landinstellingen kunnen gebruikmaken van dezelfde taal. en_US en en_GB (Groot-Brittannië) zijn bijvoorbeeld verschillende landinstellingen. In dit geval maken beide landinstellingen gebruik van de Engelse taal, maar de landcode geeft aan dat het verschillende landinstellingen betreft die daarom mogelijk gebruikmaken van verschillende assets. Bij een toepassing voor de landinstelling en_US kan het woord "kleur" bijvoorbeeld worden gespeld als "color", terwijl voor de landinstelling en_GB dit "colour" is. Ook worden valuta-eenheden aangegeven in dollars of ponden, afhankelijk van de landinstelling, en kan de notatie van datums en tijden verschillen.

U kunt ook een set assets voor een taal opgeven zonder een taalcode te specificeren. U kunt bijvoorbeeld en-assets opgeven voor de Engelse taal en aanvullende assets voor de landinstelling en_US, die specifiek zijn voor Amerikaans Engels.

Lokalisatie is meer dan het vertalen van de tekenreeksen die worden gebruikt in uw toepassing. Elk willekeurig type asset, zoals geluidsbestanden, beelden en video's, kan hierbij worden betrokken.

Een landinstelling kiezen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Om te bepalen van welke landinstelling de inhoud of toepassing gebruikmaakt, kunt u een van de volgende methoden gebruiken:

- flash.globalization-pakket: met de klassen in het flash.globalization-pakket waarmee de landinstelling kan worden vastgesteld kunt u de standaardlandinstelling voor de gebruiker ophalen op basis van het besturingssysteem en de gebruikersvoorkeuren. Dit is de voorkeursaanpak bij toepassingen die worden uitgevoerd op runtimes van Flash Player 10.1 of hoger of AIR 2.0 of hoger. Zie “De landinstelling bepalen” op pagina 999 voor meer informatie.
- Vragen aan de gebruiker - U kunt de toepassing starten met een standaardlandinstelling en dan de gebruiker vragen de gewenste landinstelling te kiezen.
- `(Alleen AIR) Capabilities.languages` - De eigenschap `Capabilities.languages` geeft een array weer met de voorkeurstalen die zijn ingesteld via het besturingssysteem van de gebruiker. Deze strings bevatten taaltags (en script- en regiogegevens, indien van toepassing) zoals gedefinieerd in RFC4646 (http://www.ietf.org/rfc/rfc4646.txt). Bij deze tekenreeksen worden koppeltekens gebruikt als scheidingsteken (bijvoorbeeld `"en-US"` of `"ja-JP"`). Het eerste item in de geretourneerde array heeft dezelfde primaire taal-id als de eigenschap language. Als `languages[0]` bijvoorbeeld wordt ingesteld op `"en-US"`, wordt de eigenschap `language` ingesteld op `"en"`. Als de taaleigenschap echter op `'xu'` is ingesteld (waarmee u een onbekende taal opgeeft), is het eerste element in de array `languages` anders.

- `Capabilities.language` - De eigenschap `Capabilities.language` geeft de taalcode voor de gebruikersinterface van het besturingssysteem op. Deze eigenschap is echter beperkt tot 20 bekende talen. Op Engelse systemen retourneert deze eigenschap alleen de taalcode, niet de landcode. Om deze redenen is het beter om het eerste element van de array `Capabilities.languages` te gebruiken.

Flex-inhoud lokaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Adobe Flex biedt een raamwerk voor het lokaliseren van Flex-inhoud. Dit raamwerk omvat de klassen Locale, ResourceBundle en ResourceManagerImpl, evenals de interfaces IResourceBundle en IResourceManagerImpl.

Er is een Flex-localisatiebibliotheek met hulpprogrammaklassen voor het sorteren van landinstellingen voor toepassingen beschikbaar op Google Code (http://code.google.com/p/as3localelib/).

Meer Help-onderwerpen

http://code.google.com/p/as3localelib/

Flash-inhoud lokaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Adobe Flash Professional bevat een Locale-klasse in de ActionScript 3.0-componenten. Met de klasse Locale kunt u bepalen hoe een SWF-bestand meertalige tekst weergeeft. Via het Flash-deelvenster Tekenreeksen kunt u in dynamische tekstvelden tekenreeks-id's gebruiken in plaats van letterlijke tekenreeksen. U kunt hiermee een SWF-bestand maken, waarmee tekst wordt weergegeven die vanaf een taalspecifiek XML-bestand is geladen. Zie de Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform voor informatie over het gebruik van de klasse Locale.

AIR-toepassingen lokaliseren

Adobe AIR 1.0 of hoger

De AIR SDK biedt een HTML Localization Framework (dit bevindt zich in het bestand AIRLocalizer.js). Dit framework bevat API's waardoor het werken met meerdere landinstellingen in HTML-toepassingen eenvoudiger wordt. Er is een ActionScript-bibliotheek voor het sorteren van landinstellingen beschikbaar op http://code.google.com/p/as3localelib/.

Meer Help-onderwerpen

http://code.google.com/p/as3localelib/

Datums, tijden en valuta's lokaliseren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De manier waarop toepassingen datums, tijden en valuta's presenteren, is heel erg afhankelijk van de landinstelling. In de Verenigde Staten wordt bijvoorbeeld voor de notatie van datums maand/dag/jaar gebruikt, terwijl de Europese standaard dag/maand/jaar is.

U kunt code schrijven om datums, tijden en valuta's op te maken. De volgende code converteert bijvoorbeeld een Date-object naar de notatie maand/dag/jaar of dag/maand/jaar. Als de variabele `locale` (die de landinstelling vertegenwoordigt) wordt ingesteld op `"en_US"`, retourneert de functie de notatie maand/dag/jaar. Het voorbeeld converteert een Date-object bij alle andere landinstellingen naar de notatie dag/maand/jaar:

```
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

Het Flex-framework bevat besturingselementen voor het opmaken van datums, tijden en valuta's. Deze besturingselementen omvatten de besturingselementen DateFormatter en CurrencyFormatter.

- mx:DateFormatter
- mx:CurrencyFormatter

Hoofdstuk 58: Informatie over de HTML-omgeving

Adobe AIR 1.0 of hoger

Adobe® AIR® gebruikt WebKit *(www.webkit.org)*, worden ook gebruikt door de Safari webbrowser om HTML- en JavaScript-inhoud te parseren, renderen en de lay-out te maken. Het gebruik van de AIR API's in HTML-inhoud is optioneel. U kunt de inhoud van een HTMLLoader-object of HTML-venster geheel in HTML en JavaScript programmeren. De meeste bestaande HTML-pagina's en -toepassingen kunnen normaal gesproken met weinig wijzigingen worden uitgevoerd (mits ze gebruik maken van HTML-, CSS-, DOM- en JavaScript-functies die compatibel zijn met WebKit).

Let op: Nieuwe versies van runtime voor Adobe AIR kunnen bijgewerkte versies van WebKit bevatten. Een WebKit-update in een nieuwe versie van AIR *kan* onverwachte wijzigingen opleveren in een geïmplementeerde AIR-toepassing. Deze wijzigingen kunnen het gedrag of de vormgeving van HTML-inhoud in een toepassing beïnvloeden. Verbeteringen of correcties in WebKit-rendering kunnen de lay-out van onderdelen in een gebruikersinterface van een toepassing wijzigen. Daarom raden we u aan om een updatemechanisme in uw toepassing te voorzien. Als u uw toepassing moet wijzigen als gevolg van een wijziging in de WebKit-versie van AIR, kan het AIR-updatemechanisme de gebruiker vragen om een nieuwe versie van uw toepassing te installeren.

In de volgende tabel ziet u de versie van de Safari-webbrowser die de WebKit-versie gebruikt die equivalent is aan de versie die in AIR wordt gebruikt:

AIR-versie	Safari-versie
1.0	2.04
1.1	3.04
1.5	4.0 Bèta
2.0	4.03
2.5	4.03
2.6	4.03
2.7	4.03
3	5.0.3

U kunt altijd controleren welke versie van WebKit is geïnstalleerd door de standaardtekenreeks voor de gebruikersagent te bekijken die wordt geretourneerd door een HTMLLoader-object:

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

Houd er rekening mee dat de WebKit-versie die in AIR wordt gebruikt, niet gelijk is aan de opensourceversie. Sommige functies worden niet ondersteund door AIR en in de AIR-versie kunnen functie- en beveiligingsproblemen zijn opgelost, terwijl dit nog niet het geval is voor de overeenkomstige WebKit-versie. Zie "WebKit-functies die niet worden ondersteund in AIR" op pagina 1034.

Aangezien AIR-toepassingen rechtstreeks op de desktopcomputer worden uitgevoerd, met volledige toegang tot het bestandssysteem, is het beveiligingsmodel voor HTML-inhoud strenger dan het beveiligingsmodel van een typische webbrowser. In AIR wordt alleen inhoud die wordt geladen uit de installatiemap van de toepassing geplaatst in de *toepassingssandbox*. De toepassingssandbox heeft het hoogste machtigingsniveau en biedt toegang tot de AIR API's. AIR plaatst andere inhoud in geïsoleerde sandboxen op basis van de locatie waar deze inhoud vandaan komt. Bestanden die zijn geladen uit het bestandssysteem, gaan naar een lokale sandbox. Bestanden die zijn geladen van het netwerk met behulp van het protocol http: of https:, gaan naar een sandbox op basis van het domein van de externe server. Inhoud in deze niet-toepassingssandboxen krijgt geen toegang tot AIR API's en wordt uitgevoerd op dezelfde manier als bij een normale webbrowser.

HTML-inhoud in AIR geeft geen SWF- of PDF-inhoud weer wanneer alfa-, schaal- of transparantie-instellingen worden toegepast. Zie "Belangrijke opmerkingen voor het laden van SWF- of PDF-inhoud in een HTML-pagina" op pagina 1065 en "Venstertransparantie" op pagina 944 voor meer informatie.

Meer Help-onderwerpen

Webkit DOM-naslaggids

Safari HTML-naslaggids

Safari CSS-naslaggids

www.webkit.org

Overzicht van de HTML-omgeving

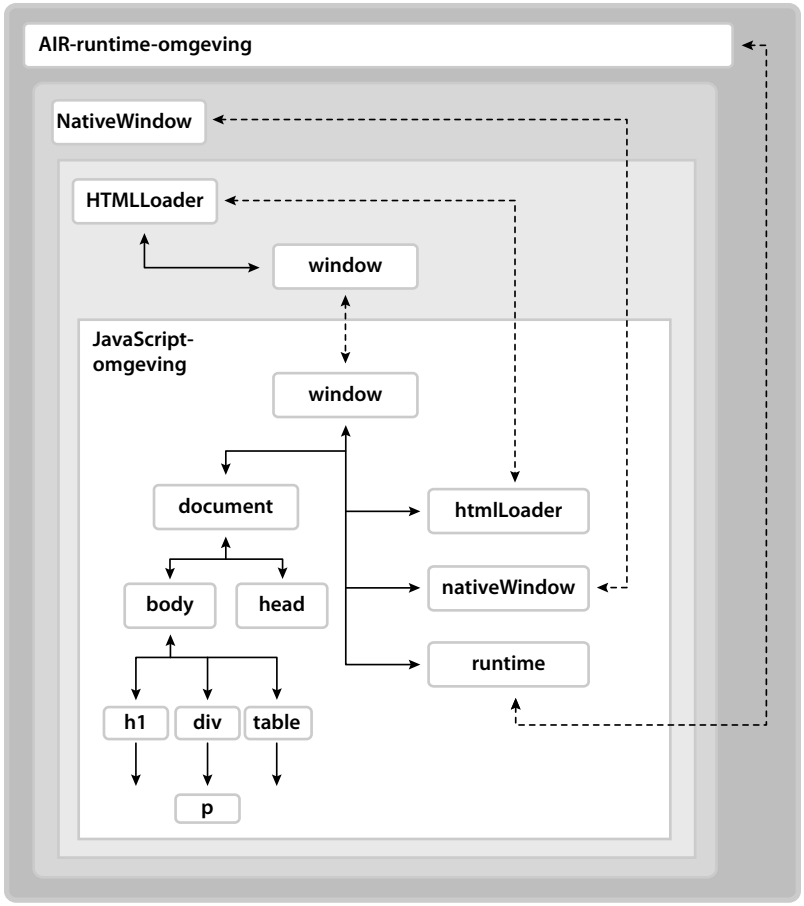
Adobe AIR 1.0 of hoger

Adobe AIR biedt een volledige browserachtige JavaScript-omgeving met een HTML-renderer, documentobjectmodel en JavaScript-interpreter. De JavaScript-omgeving wordt vertegenwoordigd door de AIR-klasse HTMLLoader. In HTML-vensters bevat een HTMLLoader-object alle HTML-inhoud. Dit object bevindt zich op zijn beurt weer in een NativeWindow-object. In SWF-inhoud kan de klasse HTMLLoader, die een uitbreiding vormt van de klasse Sprite, worden toegevoegd aan de weergavelijst van een werkgebied, net als ieder ander weergaveobject. De Adobe® ActionScript® 3.0-eigenschappen van de klasse worden beschreven in "Scripting van de AIR HTML-container" op pagina 1063 en in de *Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform*. In het Flex-framework wordt de AIR-klasse HTMLLoader opgenomen in een mx:HTML-component. De mx:HTML-component vormt een uitbreiding van de klasse UIComponent, zodat deze rechtstreeks kan worden gebruikt met andere Flex-containers. Verder is de JavaScript-omgeving binnen de mx:HTML-component identiek.

Informatie over de JavaScript-omgeving en zijn relatie met de AIR-host.

Adobe AIR 1.0 of hoger

In het volgende diagram wordt de relatie geïllustreerd tussen de JavaScript-omgeving en de AIR-runtimeomgeving. Er wordt maar één native venster weergegeven. Een AIR-toepassing kan echter meerdere vensters bevatten. (En een venster kan meerdere HTMLLoader-objecten bevatten.)



De JavaScript-omgeving heeft zijn eigen Document- en Window-objecten. JavaScript-code kan interageren met de AIR-runtimeomgeving via de eigenschappen runtime, nativeWindow en htmlLoader. ActionScript-code kan interageren met de JavaScript-omgeving via de eigenschap window van een HTMLLoader-object, dat een referentie vormt naar het JavaScript Window-object. Verder kunnen zowel ActionScript- als JavaScript-objecten luisteren naar gebeurtenissen die zijn verzonden door zowel AIR- als JavaScript-objecten.

De eigenschap `runtime` geeft toegang tot AIR API-klassen zodat u nieuwe AIR-objecten en ook toegangsklasseleden (statische leden) kunt maken. Als u een AIR API wilt openen, voegt u de naam van de klasse (met pakket) toe aan de eigenschap `runtime`. Als u bijvoorbeeld een File-object wilt maken, gebruikt u de volgende opdracht:

```
var file = new window.runtime.filesystem.File();
```

Opmerking: *De AIR SDK bevat een JavaScript-bestand, `AIRAliases.js`, waarin aanvullende handige aliassen voor de meestgebruikte AIR-klassen worden gedefinieerd. Als u dit bestand importeert, kunt u de korte vorm air.Class gebruiken in plaats van window.runtime.package.Class. U kunt bijvoorbeeld het File-object maken met `new air.File()`.*

Het NativeWindow-object bevat eigenschappen voor het beheer van het bureaubladvenster. Vanuit een HTML-pagina kunt u toegang krijgen tot het omvattende NativeWindow-object met behulp van de eigenschap `window.nativeWindow`.

Het HTMLLoader-object bevat eigenschappen, methoden en gebeurtenissen voor het beheer van de manier waarop inhoud wordt geladen en weergegeven. Vanuit een HTML-pagina kunt u toegang krijgen tot het bovenliggende HTMLLoader-object met behulp van de eigenschap `window.htmlLoader`.

Belangrijk: *Alleen pagina's die zijn geïnstalleerd als onderdeel van een toepassing, hebben de eigenschap* `htmlLoader`*,* `nativeWindow` *of* `runtime`*. Ze hebben deze eigenschappen alleen wanneer ze zijn geladen als document van het hoogste niveau. Deze eigenschappen worden niet toegevoegd wanneer een document wordt geladen in een frame of iframe. (Een onderliggend document kan toegang krijgen tot deze eigenschappen van het bovenliggende document, zolang het onderliggende document zich bevindt in dezelfde beveiligingssandbox. Een document dat bijvoorbeeld is geladen in een frame, kan toegang krijgen tot de eigenschap* `runtime` *van het bovenliggende document met behulp van* `parent.runtime`*.)*

Informatie over beveiliging

Adobe AIR 1.0 of hoger

AIR voert alle code binnen een beveiligingssandbox uit op basis van het brondomein. Toepassingsinhoud, die beperkt is tot inhoud die is geladen uit de installatiemap van de toepassing, wordt geplaatst in de *toepassings*sandbox. Toegang tot de runtimeomgeving en de AIR API's is alleen beschikbaar voor HTML en JavaScript die worden uitgevoerd binnen deze sandbox. Tegelijkertijd wordt de meeste dynamische evaluatie en uitvoering van JavaScript geblokkeerd in de toepassingssandbox nadat alle handlers voor de paginagebeurtenis `load` zijn geretourneerd.

U kunt een toepassingspagina toewijzen in een niet-toepassingssandbox door de pagina te laden in een frame of iframe en de AIR-specifieke kenmerken `sandboxRoot` en `documentRoot` van het frame in te stellen. Door de waarde `sandboxRoot` in te stellen op een werkelijk extern domein, kunt u de sandboxinhoud inschakelen zodat u inhoud in dat domein kunt cross-scripten. Het toewijzen van pagina's op deze manier kan nuttig zijn bij het laden en scripten van externe inhoud, bijvoorbeeld in een *mash-up*-toepassing.

Een andere manier om toepassings- en niet-toepassingsinhoud in staat te stellen om elkaar te cross-scripten, en de enige manier om niet-toepassingsinhoud toegang te geven tot AIR API's, is het maken van een *sandboxbridge*. Een *bovenliggende-onderliggende* bridge maakt het mogelijk voor inhoud in een onderliggend frame, iframe of venster om toegang te krijgen tot speciaal aangewezen methoden en eigenschappen die zijn gedefinieerd in de toepassingssandbox. Een *onderliggende-bovenliggende* bridge daarentegen maakt het toepassingsinhoud mogelijk om toegang te krijgen tot bepaalde methoden en eigenschappen die zijn gedefinieerd in de sandbox van de onderliggende. Sandboxbridges worden gecreëerd door de eigenschappen `parentSandboxBridge` en `childSandboxBridge` van het Window-object in te stellen. Zie "HTML-beveiliging in Adobe AIR" op pagina 1149 en"Frame- en iframe-elementen in HTML" op pagina 1030 voor meer informatie.

Informatie over invoegtoepassingen en ingesloten objecten

Adobe AIR 1.0 of hoger

AIR ondersteunt de invoegtoepassing Adobe ® Acrobat®. Gebruikers moeten beschikken over Acrobat of Adobe® Reader® 8.1 (of hoger) om PDF-inhoud te kunnen weergeven. Het HTMLLoader-object biedt een eigenschap waarmee kan worden gecontroleerd of het systeem van een gebruiker PDF kan weergeven. SWF-bestandsinhoud kan ook worden weergegeven binnen de HTML-omgeving. Deze functie is echter ingebouwd in AIR en maakt geen gebruik van een externe invoegtoepassing.

In AIR worden andere WebKit-invoegtoepassingen niet ondersteund.

Meer Help-onderwerpen

"HTML-beveiliging in Adobe AIR" op pagina 1149

"Sandboxen in HTML" op pagina 1022

"Frame- en iframe-elementen in HTML" op pagina 1030

"JavaScript-object Window" op pagina 1028

"XMLHttpRequest-objecten" op pagina 1024

"PDF-inhoud toevoegen in AIR" op pagina 581

AIR en WebKit

Adobe AIR 1.0 of hoger

Adobe AIR gebruikt de WebKit-engine voor open bronnen, die ook in de Safari-webbrowser worden gebruikt. AIR voegt verschillende extensies toe waarmee toegang tot de runtimeklassen en -objecten mogelijk is. Deze extensies hebben ook een functie bij de beveiliging. Webkit voegt zelf functies toe die niet in de W3C-standaarden voor HTML, CSS en JavaScript zitten.

Hier worden alleen AIR-uitbreidingen en de belangrijkste WebKit-extensies beschreven. Zie www.webkit.org en developer.apple.com voor meer documentatie over niet-standaard HTML, CSS en JavaScript. Zie de W3C-website voor informatie over standaarden. Mozilla biedt ook een handige algemene referentie over HTML-, CSS-, en DOM-onderwerpen (de WebKit- en Mozilla-engines zijn natuurlijk niet identiek).

JavaScript in AIR

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

AIR brengt een aantal wijzigingen aan in het normale gedrag van veelgebruikte JavaScript-objecten. Veel van deze wijzigingen worden aangebracht om het gemakkelijker te maken veilige toepassingen in AIR te schrijven. Deze verschillen in gedrag betekenen echter ook dat bepaalde veelgebruikte JavaScript-coderingspatronen en bestaande webtoepassingen die gebruikmaken van die patronen, niet altijd zoals verwacht zullen worden uitgevoerd in AIR. Zie "JavaScript-beveiligingsfouten voorkomen" op pagina 1041 voor meer informatie over het corrigeren van dit soort problemen.

Sandboxen in HTML

Adobe AIR 1.0 of hoger

AIR plaatst inhoud in geïsoleerde sandboxen afhankelijk van de oorsprong van de inhoud. De sandboxregels zijn consistent met het zelfde-oorsprongbeleid dat wordt geïmplementeerd door de meeste webbrowsers. Ze zijn ook consistent met de regels voor sandboxen die worden geïmplementeerd door Adobe Flash Player. Verder biedt AIR het nieuwe sandboxtype *toepassing*, waarmee toepassingsinhoud kan worden opgeslagen en beveiligd. Zie "Beveiligingssandboxen" op pagina 1108 voor meer informatie over de types sandboxen die u tegen kunt komen tijdens het ontwikkelen van AIR-toepassingen.

Toegang tot de runtimeomgeving en de AIR API's is alleen beschikbaar voor HTML en JavaScript die worden uitgevoerd binnen de toepassingssandbox. Tegelijkertijd wordt dynamische evaluatie en uitvoering van de verschillende vormen van JavaScript uit veiligheidsoverwegingen voor een groot deel beperkt binnen de toepassingssandbox. Deze beperkingen gelden altijd, ongeacht of de toepassing rechtstreeks informatie van een server laadt. (Zelfs bestandsinhoud, geplakte tekenreeksen en gegevens die rechtstreeks door de gebruiker worden ingevoerd, kunnen onbetrouwbaar zijn.)

De oorsprong van de inhoud in een pagina bepaalt de sandbox waaraan deze wordt toegewezen. Alleen inhoud die is geladen vanuit de toepassingsmap (de installatiemap waarnaar wordt verwezen door het URL-schema `app:`) wordt in de toepassingssandbox geplaatst. Inhoud die wordt geladen vanaf het bestandssysteem wordt geplaatst in het *Lokaal-met-bestandssysteem* of de sandbox *Lokaal-vertrouwd*, waarmee u toegang en interactie krijgt met inhoud op het lokale bestandssysteem, maar niet met de inhoud op afstand. Inhoud die wordt geladen vanaf het netwerk, wordt in een externe sandbox geplaatst die correspondeert met het brondomein.

Als een toepassingspagina toestemming moet hebben om desgewenst te interageren met inhoud in een externe sandbox, kan die pagina worden toegewezen aan hetzelfde domein als de externe inhoud. Als u bijvoorbeeld een toepassing schrijft waarin kaartgegevens uit een internetservice worden weergegeven, kan de pagina van deze toepassing waarin inhoud van de service wordt geladen en weergegeven, worden toegewezen aan het servicedomein. De kenmerken voor het toewijzen van pagina's aan externe sandboxen en domeinen zijn nieuwe kenmerken die worden toegevoegd aan de frame- en iframe-elementen in HTML.

Als u inhoud in een niet-toepassingssandbox wilt toestaan om veilig AIR-functies te gebruiken, kunt u een bovenliggende sandboxbridge instellen. Als u toepassingsinhoud wilt toestaan om veilig methoden en toegangseigenschappen van inhoud in andere sandboxen te benaderen, kunt u een onderliggende sandboxbridge instellen. Veilig betekent hier dat externe inhoud niet per ongeluk verwijzingen kan ophalen naar objecten, eigenschappen of methoden die niet expliciet beschikbaar zijn gemaakt. Alleen eenvoudige gegevenstypen, functies en anonieme objecten kunnen via de bridge worden doorgegeven. U moet echter nog steeds voorkomen dat potentieel gevaarlijke functies expliciet beschikbaar worden gemaakt. Als u bijvoorbeeld een interface beschikbaar hebt gemaakt waarmee externe inhoud bestanden op iedere willekeurige locatie in het systeem van een gebruiker kan lezen en schrijven, biedt u externe inhoud mogelijk kans om uw gebruikers grote schade te berokkenen.

De JavaScript-functie eval()

Adobe AIR 1.0 of hoger

Het gebruik van de functie `eval()` is beperkt binnen de toepassingssandbox als een pagina eenmaal is geladen. Sommige vormen van gebruik zijn toegestaan, zodat gegevens met JSON-indeling veilig kunnen worden geparseerd. Een evaluatie die uitvoerbare instructies als gevolg heeft, resulteert echter altijd in een fout. “Codebeperkingen voor de inhoud van verschillende sandboxen” op pagina 1152 beschrijft het toegestane gebruik van de functie `eval()`.

Functieconstructors

Adobe AIR 1.0 of hoger

In de toepassingssandbox kunnen functieconstructors worden gebruikt voordat een pagina geheel is geladen. Nadat alle handlers voor de paginagebeurtenis `load` zijn voltooid, kunnen geen nieuwe functies meer worden gemaakt.

Externe scripts laden

Adobe AIR 1.0 of hoger

HTML-pagina's in de toepassingssandbox kunnen geen `script`tags gebruiken om JavaScript-bestanden te laden van buiten de toepassingsmap. Als een pagina in uw toepassing een script van buiten de toepassingsmap moet laden, moet die pagina worden toegewezen aan een niet-toepassingssandbox.

XMLHttpRequest-objecten

Adobe AIR 1.0 of hoger

AIR is voorzien van een XHR-object (XMLHttpRequest), waarmee toepassingen gegevens kunnen aanvragen. In het volgende voorbeeld ziet u een eenvoudige gegevensaanvraag:

```
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);
```

In tegenstelling tot een browser is het bij AIR mogelijk dat inhoud die wordt uitgevoerd in de toepassingssandbox, gegevens uit elk willekeurig domein aanvraagt. Het resultaat van een XHR die een JSON-tekenreeks bevat, kan worden geëvalueerd in gegevensobjecten, behalve als het resultaat ook uitvoerbare code bevat. Als het XHR-resultaat ook uitvoerbare instructies bevat, treedt er een fout op en mislukt de evaluatiepoging.

Om te voorkomen dat code uit externe bronnen per ongeluk wordt ingebracht, retourneren synchrone XHR's een leeg resultaat als ze zijn gemaakt voordat een pagina geheel is geladen. Asynchrone XHR's worden altijd geretourneerd nadat een pagina is geladen.

AIR blokkeert standaard domeinoverschrijdende XMLHttpRequests in niet-toepassingssandboxen. Een bovenliggend venster in de toepassingssandbox kan ervoor kiezen domeinoverschrijdende aanvragen in een onderliggend frame dat inhoud in een niet-toepassingssandbox bevat, toe te staan door het kenmerk `allowCrossDomainXHR`, dat wordt toegevoegd door AIR, in te stellen op `true` in het omvattende frame- of iframe-element:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

Opmerking: *Desgewenst kan de AIR-klasse URLStream ook worden gebruikt om gegevens te downloaden.*

Als u een XMLHttpRequest naar een externe server stuurt vanuit een frame of iframe dat toepassingsinhoud bevat die is toegewezen aan een externe sandbox, moet u ervoor zorgen dat de toewijzende URL het serveradres dat in de XHR wordt gebruikt, niet maskeert. Bekijk bijvoorbeeld de volgende iframe-definitie, die toepassingsinhoud toewijst in een externe sandbox voor het domein example.com:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

Omdat het kenmerk `sandboxRoot` de basis-URL van het adres www.example.com opnieuw toewijst, worden alle aanvragen geladen vanuit de toepassingsmap en niet vanaf de externe server. Aanvragen worden altijd opnieuw toegewezen, ongeacht of ze afkomstig zijn van paginanavigatie of een XMLHttpRequest.

Om te voorkomen dat gegevensaanvragen naar de externe server per ongeluk worden geblokkeerd, wijst u `sandboxRoot` toe aan een submap van de externe URL en niet aan de hoofdmap. Die map hoeft niet echt te bestaan. Als u bijvoorbeeld wilt toestaan dat aanvragen naar www.example.com gegevens laden vanaf de externe server en niet uit de toepassingsmap, verandert u het vorige iframe-element in de volgende:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

In dit geval wordt alleen de inhoud van de submap `air` lokaal geladen.

Zie "Frame- en iframe-elementen in HTML" op pagina 1030 en "HTML-beveiliging in Adobe AIR" op pagina 1149 voor meer informatie over sandboxtoewijzing.

Cookies

Adobe AIR 1.0 of hoger

In AIR-toepassingen kan alleen inhoud in externe sandboxen (inhoud die is geladen vanaf bronnen van het type http: en https:) gebruikmaken van cookies (de eigenschap `document.cookie`). In de toepassingssandbox, zijn andere opslagmethoden beschikbaar voor het opslaan van persistente gegevens, zoals de klassen EncryptedLocalStore, SharedObject en FileStream.

Clipboard-objecten

Adobe AIR 1.0 of hoger

De WebKit-API voor Clipboard wordt aangestuurd door de volgende gebeurtenissen: `copy`, `cut` en `paste`. Het gebeurtenisobject dat bij deze gebeurtenissen wordt doorgegeven, geeft via de eigenschap `clipboardData` toegang tot het klembord. Gebruik de volgende methoden van het object `clipboardData` om klembordgegevens te lezen of te schrijven:

Methode	Beschrijving
clearData(mimeType)	Hiermee wist u de klembordgegevens. Stel de parameter `mimeType` in op het MIME-type van de gegevens die u wilt wissen.
getData(mimeType)	Hiermee haalt u de klembordgegevens op. Deze methode kan alleen worden opgeroepen in een handler voor de gebeurtenis `paste`. Stel de parameter `mimeType` in op het MIME-type van de gegevens die u wilt retourneren.
setData(mimeType, data)	Hiermee kopieert u gegevens naar het klembord. Stel de parameter `mimeType` in op het MIME-type van de gegevens.

JavaScript-code buiten de toepassingssandbox kan alleen via deze gebeurtenissen toegang krijgen tot het klembord. Inhoud in de toepassingssandbox kan echter rechtstreeks toegang krijgen tot het systeemklembord via de AIR-klasse Clipboard. U kunt bijvoorbeeld de volgende opdracht gebruiken om gegevens over tekstopmaak op het klembord te krijgen:

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                    air.ClipboardTransferMode.ORIGINAL_ONLY);
```

De volgende MIME-typen zijn geldig voor gegevens:

MIME-type	Waarde
Tekst	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Bitmap	"image/x-vnd.adobe.air.bitmap"
Bestandenlijst	"application/x-vnd.adobe.air.file-list"

Belangrijk: *Alleen inhoud in de toepassingssandbox kan toegang krijgen tot bestandsgegevens die aanwezig zijn op het klembord. Als niet-toepassingsinhoud probeert toegang te krijgen tot een bestandsobject vanaf het klembord, wordt een beveiligingsfout veroorzaakt.*

Zie "Kopiëren en plakken" op pagina 629 en Using the Pasteboard from JavaScript (Het klembord voor plakken in JavaScript gebruiken) in het Apple Developer Center voor meer informatie over het gebruik van het klembord.

Slepen en neerzetten

Adobe AIR 1.0 of hoger

Het slepen en neerzetten in en uit HTML produceert de volgende DOM-gebeurtenissen: `dragstart`, `drag`, `dragend`, `dragenter`, `dragover`, `dragleave` en `drop`. Het gebeurtenisobject dat bij deze gebeurtenissen wordt doorgegeven, geeft via de eigenschap `dataTransfer` toegang tot de gesleepte gegevens. De eigenschap `dataTransfer` verwijst naar een object dat dezelfde methoden biedt als het `clipboardData`-object dat is gekoppeld aan een gebeurtenis van het type clipboard. U kunt bijvoorbeeld de volgende functie gebruiken om tekstopmaakgegevens van de gebeurtenis `drop` te krijgen:

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

Het `dataTransfer`-object heeft de volgende belangrijke leden:

Lid	Beschrijving
clearData(mimeType)	Hiermee wist u de gegevens. Stel de parameter `mimeType` in op het MIME-type van de gegevensrepresentatie die u wilt wissen.
getData(mimeType)	Hiermee haalt u de gesleepte gegevens op. Deze methode kan alleen worden opgeroepen in een handler voor de gebeurtenis `drop`. Stel de parameter `mimeType` in op het MIME-type van de gegevens die u wilt ophalen.
setData(mimeType, data)	Hiermee stelt u de gegevens in die moeten worden gesleept. Stel de parameter `mimeType` in op het MIME-type van de gegevens.

Lid	Beschrijving
types	Een array van tekenreeksen die de MIME-typen bevat van alle gegevensrepresentaties die nu beschikbaar zijn in het `dataTransfer`-object.
effectsAllowed	Hiermee geeft u aan of de gesleepte gegevens kunnen worden gekopieerd, verplaatst, gekoppeld of een combinatie daarvan. Stel de eigenschap `effectsAllowed` in de handler voor de gebeurtenis `dragstart` in.
dropEffect	Hiermee geeft u aan welke van de toegestane neerzeteffecten (drop) worden ondersteund door een sleepdoel (drag). Stel de eigenschap `dropEffect` in de handler voor de gebeurtenis `dragEnter` in. Tijdens het slepen verandert de cursor van vorm. Deze geeft aan wat voor effect er zou optreden als de gebruiker de muis loslaat. Als er geen `dropEffect` is opgegeven, wordt een eigenschapseffect uit `effectsAllowed` gekozen. Het kopieereffect (copy) heeft prioriteit over het verplaatsingseffect (move), en dat heeft weer prioriteit over het koppelingseffect (link). De gebruiker kan de standaardprioriteit wijzigen met behulp van het toetsenbord.

Zie "Slepen en neerzetten in AIR" op pagina 642 en Slepen en neerzetten gebruiken JavaScript (Apple Developer Center) voor meer informatie over het toevoegen van ondersteuning voor slepen en neerzetten aan een AIR-toepassing.

De eigenschappen innerHTML en outerHTML

Adobe AIR 1.0 of hoger

AIR stelt beveiligingsbeperkingen in voor het gebruik van de eigenschappen `innerHTML` en `outerHTML` voor het uitvoeren van inhoud in de toepassingssandbox. Voordat de paginagebeurtenis load optreedt en terwijl eventuele gebeurtenishandlers voor load worden uitgevoerd, gelden er geen beperkingen voor het gebruik van de eigenschappen `innerHTML` en `outerHTML`. Als de pagina echter is geladen, kunt u de eigenschap `innerHTML` of `outerHTML` alleen gebruiken om statische inhoud toe te voegen aan het document. Eventuele opdrachten die zich bevinden in de tekenreeks die is toegewezen aan `innerHTML` of `outerHTML` en die worden geëvalueerd naar uitvoerbare code, worden genegeerd. Als u bijvoorbeeld een kenmerk om een gebeurtenis terug te roepen (callback) opneemt in een elementdefinitie, wordt de gebeurtenislistener niet toegevoegd. Op dezelfde manier worden ingesloten tags van het type `<script>` niet geëvalueerd. Zie "HTML-beveiliging in Adobe AIR" op pagina 1149 voor meer informatie.

De methoden Document.write() en Document.writeln()

Adobe AIR 1.0 of hoger

Het gebruik van de methoden `write()` en `writeln()` is niet beperkt in de toepassingssandbox vóór de paginagebeurtenis `load`. Als de pagina echter is geladen, wordt bij het oproepen van een van deze methoden de pagina niet gewist en wordt er ook geen nieuwe pagina gemaakt. In een niet-toepassingssandbox wordt, net zoals bij de meeste webbrowsers, door het oproepen van `document.write()` of `writeln()` nadat een pagina geheel is geladen, de huidige pagina gewist en wordt een nieuwe lege pagina geopend.

De eigenschap Document.designMode

Adobe AIR 1.0 of hoger

Hiermee stelt u de eigenschap `document.designMode` in op de waarde `on` als u alle elementen in het document bewerkbaar wilt maken. Met de ingebouwde editor kunt u tekst bewerken, kopiëren, plakken, slepen en neerzetten. Het instellen van `designMode` op `on` is equivalent met het instellen van de eigenschap `contentEditable` van het element `body` op `true`. U kunt de eigenschap `contentEditable` voor de meeste HTML-elementen gebruiken om te definiëren welke secties van een document kunnen worden bewerkt. Zie "Het HTML-kenmerk contentEditable" op pagina 1033 voor meer informatie.

unload-gebeurtenissen (voor body- en frameset-objecten)

Adobe AIR 1.0 of hoger

In de tag `frameset` of `body` van het bovenste niveau van een venster (inclusief het hoofdvenster van de toepassing), mag u de gebeurtenis `unload` niet gebruiken om te reageren op het venster dat of de toepassing die wordt gesloten. In plaats daarvan gebruikt u de gebeurtenis `exiting` van het NativeApplication-object (om te bepalen wanneer een toepassing wordt gesloten). U kunt ook de gebeurtenis `closing` van het NativeApplication-object gebruiken (om te bepalen wanneer een toepassing wordt gesloten). De volgende JavaScript-code geeft bijvoorbeeld het bericht `"Goodbye"` weer wanneer de gebruiker de toepassing sluit:

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

Scripts kunnen echter *wel* succesvol reageren op de gebeurtenis `unload` die wordt veroorzaakt door navigatie van een frame, iframe of vensterinhoud van het bovenste niveau.

Opmerking: *In een volgende versie van Adobe AIR worden deze beperkingen mogelijk verwijderd.*

JavaScript-object Window

Adobe AIR 1.0 of hoger

Het Window-object blijft het globale object binnen de JavaScript-uitvoeringscontext. In de toepassingssandbox voegt AIR nieuwe eigenschappen toe aan het JavaScript-object Window ten einde toegang te geven tot de ingebouwde AIR-klassen en belangrijke hostobjecten. Verder vertonen bepaalde methoden en eigenschappen binnen de toepassingssandbox ander gedrag dan daarbuiten.

De eigenschap Window.runtime De eigenschap `runtime` stelt u in staat om de ingebouwde runtimeklassen te instantiëren en te gebruiken vanuit de toepassingssandbox. Deze klassen omvatten de AIR- en Flash Player-API's (maar bijvoorbeeld niet het Flex-framework). Met de volgende opdracht maakt u bijvoorbeeld een AIR-bestandsobject:

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

Het bestand `AIRAliases.js`, dat zich bevindt in de AIR SDK, bevat aliasdefinities waarmee u dergelijke verwijzingen kunt verkorten. Als `AIRAliases.js` bijvoorbeeld in een pagina wordt geïmporteerd, kan een File-object worden gemaakt met de volgende opdracht:

```
var preferencesFile = new air.File();
```

De eigenschap `window.runtime` is alleen gedefinieerd voor inhoud binnen de toepassingssandbox en alleen voor het bovenliggende document van een pagina met frames of iframes.

Zie "Het bestand AIRAliases.js gebruiken" op pagina 1047.

De eigenschap Window.nativeWindow De eigenschap `nativeWindow` biedt een verwijzing naar het onderliggende native Window-object. Met deze eigenschap kunt u vensterfuncties en eigenschappen zoals schermpositie, grootte en zichtbaarheid in een script opnemen. Ook kunt u venstergebeurtenissen zoals sluiten, grootte wijzigen en verplaatsen hiermee verwerken. Met de volgende opdracht sluit u bijvoorbeeld het venster:

```
window.nativeWindow.close();
```

Opmerking: *De functies voor vensterbesturing die worden geboden door het NativeWindow-object, overlappen de functies die worden geboden door het JavaScript-object Window. In dergelijke gevallen kunt u de methode gebruiken die u het handigst vindt.*

De eigenschap `window.nativeWindow` is alleen gedefinieerd voor inhoud binnen de toepassingssandbox en alleen voor het bovenliggende document van een pagina met frames of iframes.

De eigenschap Window.htmlLoader De eigenschap `htmlLoader` biedt een referentie voor het AIR-object HTMLLoader dat de HTML-inhoud bevat. Met deze eigenschap kunt u het uiterlijk en het gedrag van de HTML-omgeving in een script opnemen. U kunt bijvoorbeeld de eigenschap `htmlLoader.paintsDefaultBackground` gebruiken om te bepalen of het besturingselement een standaard witte achtergrond gebruikt:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Opmerking: *Het HTMLLoader-object zelf heeft een eigenschap* `window` *die verwijst naar het JavaScript-object Window van de HTML-inhoud die het bevat. U kunt deze eigenschap gebruiken om toegang te krijgen tot de JavaScript-omgeving via een verwijzing naar de bevattende HTMLLoader.*

De eigenschap `window.htmlLoader` is alleen gedefinieerd voor inhoud binnen de toepassingssandbox en alleen voor het bovenliggende document van een pagina met frames of iframes.

De eigenschappen Window.parentSandboxBridge en Window.childSandboxBridge Met de eigenschappen `parentSandboxBridge` en `childSandboxBridge` kunt u een interface definiëren tussen een bovenliggend en een onderliggend frame. Zie "Cross-scripting van inhoud in verschillende beveiligingssandboxen" op pagina 1058 voor meer informatie.

De functies Window.setTimeout() en Window.setInterval() AIR past beveiligingsbeperkingen toe op het gebruik van de functies `setTimeout()` en `setInterval()` binnen de toepassingssandbox. U kunt de code die moet worden uitgevoerd, niet definiëren als tekenreeks bij het oproepen van `setTimeout()` of `setInterval()`. U moet hiervoor een functieverwijzing gebruiken. Zie "setTimeout() en setInterval()" op pagina 1044 voor meer informatie.

De functie Window.open() Wanneer de methode `open()` wordt opgeroepen door code die wordt uitgevoerd in een niet-toepassingssandbox, wordt er als gevolg van gebruikersinteractie (bijvoorbeeld een muisklik of het indrukken van een toets) alleen een venster geopend. Verder staat de toepassingstitel vóór de venstertitel (om te verhinderen dat vensters die worden geopend door externe inhoud, zich voordoen als vensters die worden geopend door de toepassing). Zie "Beperkingen betreffende het oproepen van de JavaScript-methode window.open()" op pagina 1155 voor meer informatie.

Het object air.NativeApplication

Adobe AIR 1.0 of hoger

Het NativeApplication-object biedt informatie over de toepassingsstatus, verzendt verschillende belangrijke gebeurtenissen op toepassingsniveau en biedt nuttige functies voor het beheer van het gedrag van de toepassing. Er wordt automatisch één instantie van het NativeApplication-object gemaakt. Deze kan worden benaderd via de door de klasse gedefinieerde eigenschap `NativeApplication.nativeApplication`.

Om vanuit JavaScript-code toegang te krijgen tot het object, kunt u het volgende gebruiken:

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

Als het script `AIRAliases.js` is geïmporteerd, kunt u ook de kortere vorm gebruiken:

```
var app = air.NativeApplication.nativeApplication;
```

U kunt het NativeApplication-object alleen benaderen vanuit de toepassingssandbox. Zie "Werken met AIR-runtime- en besturingssysteeminformatie" op pagina 937 voor meer informatie over het NativeApplication-object.

Het JavaScript URL-schema

Adobe AIR 1.0 of hoger

Uitvoering van code die is gedefinieerd in een JavaScript URL-schema (zoals in `href="javascript:alert('Test')"`), wordt geblokkeerd binnen de toepassingssandbox. Er treedt geen fout op.

HTML in AIR

Adobe AIR 1.0 of hoger

AIR en WebKit definiëren een aantal niet-standaard HTML-elementen en -kenmerken, zoals:

"Frame- en iframe-elementen in HTML" op pagina 1030

"Gebeurtenishandlers voor HTML-elementen" op pagina 1032

Frame- en iframe-elementen in HTML

Adobe AIR 1.0 of hoger

AIR voegt nieuwe kenmerken toe aan de frame- en iframe-elementen van inhoud in de toepassingssandbox:

Het kenmerk sandboxRoot Het kenmerk `sandboxRoot` geeft een alternatief, niet-toepassingsbrondomein op voor het bestand dat wordt opgegeven door het framekenmerk `src`. Het bestand wordt geladen in de niet-toepassingssandbox die correspondeert met het opgegeven domein. Inhoud in het bestand en inhoud die wordt geladen van het opgegeven domein, kan onderling cross-scripten.

Belangrijk: *Als u de waarde van `sandboxRoot` instelt op de basis-URL van het domein, worden alle aanvragen voor inhoud van dat domein geladen uit de toepassingsmap en niet vanaf de externe server (of die aanvraag nu resulteert uit paginanavigatie, uit een XMLHttpRequest of uit een andere manier om inhoud te laden).*

Het kenmerk documentRoot Het kenmerk `documentRoot` geeft de lokale map op waaruit URL's moeten worden geladen die worden omgezet in bestanden in de locatie die wordt opgegeven door `sandboxRoot`.

Bij het omzetten van URL's, of dit nu gebeurt bij het framekenmerk `src` dan wel in inhoud die in het frame wordt geladen, wordt het deel van de URL dat overeenkomt met de waarde die wordt opgegeven in `sandboxRoot`, vervangen door de waarde die wordt opgegeven in `documentRoot`. Bij de volgende frametag geldt dus:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html` wordt geladen uit de submap `sandbox` van de toepassingsinstallatiemap. Relatieve URL's in `child.html` worden omgezet op basis van de map `sandbox`. Merk op dat bestanden op de externe server op `www.example.com/air`niet toegankelijk zijn in het frame, omdat AIR zou proberen ze te laden uit de map app:/sandbox/.

Het kenmerk allowCrossDomainXHR Neem de code `allowCrossDomainXHR="allowCrossDomainXHR"` op in de openingsframetag als u wilt toestaan dat inhoud in het frame XMLHttpRequests uitvoert naar een willekeurig extern domein. Standaard kan niet-toepassingsinhoud dergelijke aanvragen alleen uitvoeren binnen het eigen brondomein. Het toestaan van XHR's naar andere domeinen heeft ernstige gevolgen voor de beveiliging. Code in de pagina kan gegevens uitwisselen met een willekeurig domein. Als er op de een of andere manier kwaadaardige inhoud in de pagina wordt ingebracht, kunnen alle gegevens die toegankelijk zijn voor code in de huidige sandbox, worden aangetast. Schakel alleen XHR's die andere domeinen kunnen bereiken in voor pagina's die u zelf maakt en beheert, en alleen als het laden van gegevens uit andere domeinen absoluut noodzakelijk is. Ook moet u alle externe gegevens die door de

pagina worden geladen zorgvuldig valideren om te voorkomen dat code wordt ingebracht of dat andere aanvallen worden uitgevoerd.

Belangrijk: *Als het kenmerk* `allowCrossDomainXHR` *is opgenomen in een frame- of iframe-element, zijn XHR's naar andere domeinen ingeschakeld (behalve als de toegewezen waarde "0" is of begint met de letter "f" of "n"). Als u bijvoorbeeld* `allowCrossDomainXHR` *instelt op* `"deny"` *(weigeren), zijn XHR's naar andere domeinen nog steeds toegestaan. Laat het kenmerk helemaal weg uit de elementdeclaratie als u aanvragen naar andere domeinen niet wilt inschakelen.*

ondominitialize, kenmerk Geeft een gebeurtenishandler op voor de framegebeurtenis `dominitialize`. Dit is een AIR-specifieke gebeurtenis die wordt geactiveerd wanneer de window- en document-objecten van het frame zijn gemaakt, maar voordat er scripts zijn geparseerd of documentelementen zijn gemaakt.

Het frame verzendt de gebeurtenis `dominitialize` zo vroeg tijdens de uitvoering van de laadsequentie, dat eventuele scripts in de onderliggende pagina kunnen verwijzen naar objecten, variabelen en functies die door de handler `dominitialize` zijn toegevoegd aan het onderliggende document. De bovenliggende pagina kan alleen rechtstreeks objecten toevoegen aan of toegang krijgen tot objecten in een onderliggend document, als de bovenliggende pagina zich bevindt in dezelfde sandbox als de onderliggende. Een bovenliggende pagina in de toepassingssandbox kan echter wel een sandboxbridge tot stand brengen voor communicatie met inhoud in een niet-toepassingssandbox.

In de volgende voorbeelden ziet u hoe u de iframe-tag in AIR gebruikt:

Plaats `child.html` in een externe sandbox zonder toewijzing aan een werkelijk domein op een externe server:

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

Plaats `child.html` in een externe sandbox, waarbij XMLHttpRequests alleen zijn toegestaan voor `www.example.com`:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

Plaats `child.html` in een externe sandbox, waarbij XMLHttpRequests zijn toegestaan voor een willekeurig extern domein:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

Plaats `child.html` in een lokaal-met-bestandssysteem sandbox:

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

Plaats `child.html` in een externe sandbox waarbij de gebeurtenis `dominitialize` wordt gebruikt om een sandboxbridge tot stand te brengen.

```
<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>
```

Het volgende document, `child.html`, illustreert hoe onderliggende inhoud de bovenliggende sandboxbridge kan benaderen:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Zie "Cross-scripting van inhoud in verschillende beveiligingssandboxen" op pagina 1058 en "HTML-beveiliging in Adobe AIR" op pagina 1149 voor meer informatie.

Gebeurtenishandlers voor HTML-elementen

Adobe AIR 1.0 of hoger

DOM-objecten in AIR en WebKit verzenden sommige gebeurtenissen die zich niet in het standaard DOM-gebeurtenismodel bevinden. In de volgende tabel worden de verwante gebeurteniskenmerken aangegeven die u kunt gebruiken om handlers voor deze gebeurtenissen op te geven:

Naam callbackkenmerk	Beschrijving
oncontextmenu	Wordt opgeroepen wanneer een contextmenu wordt opgeroepen, bijvoorbeeld door met de rechtermuisknop op geselecteerde tekst te klikken of de Command-toets ingedrukt te houden tijdens het klikken.
oncopy	Wordt opgeroepen wanneer een selectie in een element wordt gekopieerd.
oncut	Wordt opgeroepen wanneer een selectie in een element wordt geknipt.
ondominitialize	Wordt opgeroepen wanneer het DOM wordt gemaakt van een document dat is geladen in een frame of iframe, maar voordat eventuele DOM-elementen worden gemaakt of scripts geparseerd.

Naam callbackkenmerk	Beschrijving
ondrag	Wordt opgeroepen wanneer een element wordt gesleept.
ondragend	Wordt opgeroepen wanneer een gesleept element wordt losgelaten.
ondragenter	Wordt opgeroepen wanneer een sleepbeweging de grenzen van een element bereikt.
ondragleave	Wordt opgeroepen wanneer een sleepbeweging de grenzen van een element verlaat.
ondragover	Wordt continu opgeroepen zolang een sleepbeweging zich binnen de grenzen van een element bevindt.
ondragstart	Wordt opgeroepen wanneer een sleepbeweging begint.
ondrop	Wordt opgeroepen wanneer een sleepbeweging wordt losgelaten boven een element.
onerror	Wordt opgeroepen wanneer er een fout optreedt bij het laden van een element.
oninput	Wordt opgeroepen wanneer er tekst wordt ingevoerd in een formulierelement.
onpaste	Wordt opgeroepen wanneer een item wordt geplakt in een element.
onscroll	Wordt opgeroepen wanneer de inhoud van een schuifbaar element wordt geschoven.
onselectstart	Wordt opgeroepen wanneer een selectie begint.

Het HTML-kenmerk contentEditable

Adobe AIR 1.0 of hoger

U kunt het kenmerk `contentEditable` aan een willekeurig HTML-element toevoegen om gebruikers toe te staan de inhoud van het element te bewerken. Met de volgende HTML-voorbeeldcode wordt het gehele document ingesteld als bewerkbaar, behalve het eerste `p`-element:

```
<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>
```

Opmerking: *Als u de eigenschap* `document.designMode` *instelt op* `on`*, zijn alle elementen in het document bewerkbaar, ongeacht de instelling van* `contentEditable` *voor een individueel element. Als u echter* `designMode` *instelt op* `off`*, kunnen elementen waarvoor* `contentEditable` *is ingesteld op* `true`*, nog steeds worden bewerkt. Zie "De eigenschap Document.designMode" op pagina 1027 voor meer informatie.*

URL's met 'data:'

Adobe AIR 2 of hoger

AIR biedt ondersteuning voor URL's met `data:` voor de volgende elementen:

- img
- invoertype=”image”
- CSS-regels die afbeeldingen toestaan (bijvoorbeeld 'background-image')

Met URL's met 'data:' kunt u binaire afbeeldingsgegevens direct invoegen in een CSS- of HTML-document als een base64-gecodeerde tekenreeks In het volgende voorbeeld wordt een URL met 'data:' gebruikt als een zich herhalende achtergrond:

```
<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>
```

Bij het gebruik van URL's met 'data:' is een extra witruimte relevant. De gegevenstekenreeks moet bijvoorbeeld worden ingevoerd als een enkele, niet-onderbroken regel. Anders worden de regeleinden geïnterpreteerd als onderdeel van de gegevensset en kan de afbeelding niet worden gedecodeerd.

CSS in AIR

Adobe AIR 1.0 of hoger

WebKit ondersteunt verschillende uitgebreide CSS-eigenschappen. Veel van deze uitbreidingen gebruiken het voorvoegsel: `-webkit`. Sommige van deze uitbreidingen zijn experimenteel en kunnen worden verwijderd uit een toekomstige versie van WebKit. Voor meer informatie over de Webkit-ondersteuning voor CSS en de CSS-uitbreidingen raadpleegt u de website Safari CSS-naslagids.

WebKit-functies die niet worden ondersteund in AIR

Adobe AIR 1.0 of hoger

AIR biedt geen ondersteuning voor de volgende functies van WebKit of Safari 4:

- Messaging tussen domeinen via window.postMessage (AIR beschikt over eigen API's voor communicatie tussen domeinen)

- CSS-variabelen
- WOFF (Web Open Font Format)- en SVG-lettertypen.
- HTML-video- en audiotags
- Query's voor media-apparaten
- Offline-toepassingscache
- Afdrukken (AIR bevat een eigen PrintJob-API)
- Spelling- en grammaticacontrole
- SVG
- WAI-ARIA
- WebSockets (AIR bevat eigen socket-API's)
- Webwerkprogramma's
- WebKit-SQL-API (AIR beschikt over een eigen API)
- WebKit-geolocatie-API (AIR beschikt over een eigen geolocatie-API op ondersteunde apparaten)
- WebKit-API voor het uploaden van meerdere bestanden
- WebKit-aanraakgebeurtenissen (AIR beschikt over eigen aanraakgebeurtenissen)
- WML (Wireless Markup Language)

De volgende overzichten bevatten bepaalde JavaScript-API's, HTML-elementen en CSS-eigenschappen en waarden die niet worden ondersteund door AIR:

Niet-ondersteunde JavaScript Window-objectleden:

- applicationCache()
- console
- openDatabase()
- postMessage()
- document.print()

Niet-ondersteunde HTML-tags:

- audio
- video

Niet-ondersteunde HTML-kenmerken:

- aria-*
- draggable
- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange

- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove
- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

Niet-ondersteunde JavaScript-gebeurtenissen:

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove
- touchend
- touchcancel
- webkitbeginfullscreen
- webkitendfullscreen

Niet-ondersteunde CSS-kenmerken:

- background-clip
- background-origin (gebruik -webkit-background-origin)
- background-repeat-x
- background-repeat-y
- background-size (gebruik -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

Niet-ondersteunde CSS-waarden:

- eigenschapswaarden voor weergave:
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button
- border-box (background-clip en background-origin)
- contain (background-size)
- content-box (background-clip en background-origin)
- cover (background-size)
- eigenschapswaarden voor overzicht:
  - 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 (eigenschap voor weergave)

Hoofdstuk 59: HTML en JavaScript programmeren in AIR

Adobe AIR 1.0 of hoger

Een aantal programmeringsonderwerpen zijn uniek voor het ontwikkelen van Adobe® AIR®-toepassingen met HTML en JavaScript. De volgende informatie is belangrijk, ongeacht of u een HTML-gebaseerde AIR-toepassing programmeert, of dat u een SWF-gebaseerde AIR-toepassing programmeert die HTML en JavaScript uitvoert met behulp van de klasse HTMLLoader (of de Flex™-component mx:HTML).

Informatie over de klasse HTMLLoader

Adobe AIR 1.0 of hoger

De klasse HTMLLoader van Adobe AIR definieert het weergaveobject dat HTML-inhoud kan weergeven in een AIR-toepassing. SWF-gebaseerde toepassingen kunnen een HTMLLoader-besturingselement aan een bestaand venster toevoegen of een HTML-venster maken dat automatisch een HTMLLoader-object bevat met `HTMLLoader.createRootWindow()`. Het HTMLLoader-object kan vanuit de geladen HTML-pagina worden opgeroepen met de JavaScript-eigenschap `window.htmlLoader`.

HTML-inhoud laden vanuit een URL

Adobe AIR 1.0 of hoger

De volgende code laadt een URL in een HTMLLoader-object (voeg de HTMLLoader toe als onderliggend object van de Stage of andere weergaveobjectcontainer om de HTML-inhoud in uw toepassing weer te geven):

```
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);
```

De eigenschappen `width` en `height` van een HTMLLoader-object zijn standaard allebei ingesteld op 0. U wordt aangeraden deze afmetingen in te stellen als u een HTMLLoader-object aan het Stage-object toevoegt. De HTMLLoader verstuurt verschillende gebeurtenissen terwijl een pagina wordt geladen. Op basis van deze gebeurtenissen kunt u bepalen wanneer het veilig is om met de geladen pagina te interageren. Deze gebeurtenissen worden beschreven in “Aan HTML gerelateerde gebeurtenissen in AIR afhandelen” op pagina 1082.

Opmerking: *In het Flex-framework kunnen alleen klassen die de klasse UIComponent uitbreiden, worden toegevoegd als onderliggende elementen van Flex Container-componenten. Daarom kunt u geen HTMLLoader rechtstreeks toevoegen als onderliggende component van een Flex Container-component. U kunt echter het Flex-besturingselement mx:HTML gebruiken om een aangepaste klasse te ontwikkelen die de UIComponent uitbreidt en een HTMLLoader als onderliggend element van de UIComponent bevat. U kunt de HTMLLoader ook toevoegen als onderliggend element van een UIComponent en de UIComponent toevoegen aan de Flex-container.*

U kunt HTML-tekst ook renderen met behulp van de klasse TextField, maar de mogelijkheden hiervan zijn beperkt. De klasse TextField van Adobe® Flash® Player ondersteunt een subset van de HTML-markeringen, maar vanwege de groottebeperkingen zijn de mogelijkheden hiervan beperkt. (De klasse HTMLLoader die in Adobe AIR is opgenomen, is niet beschikbaar in Flash Player.)

HTML-inhoud laden vanuit een tekenreeks

Adobe AIR 1.0 of hoger

De methode `loadString()` van een HTMLLoader-object laadt een tekenreeks met HTML-inhoud in het HTMLLoader-object:

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

Standaard wordt inhoud die is geladen via de methode `loadString()`, geplaatst in een niet-toepassingssandbox met de volgende kenmerken:

- Deze kan inhoud laden van het netwerk, maar niet van het bestandssysteem.
- Deze kan geen gegevens laden met behulp van XMLHttpRequest.
- De eigenschap `window.location` wordt ingesteld op `"about:blank"`.
- De inhoud kan geen toegang krijgen tot de eigenschap `window.runtime` (inhoud in een niet-toepassingssandbox kan dit bijvoorbeeld wel).

In AIR 1.5 omvat de klasse HTMLLoader een eigenschap `placeLoadStringContentInApplicationSandbox`. Wanneer deze eigenschap voor een HTMLLoader-object is ingesteld op `true`, wordt inhoud die wordt geladen via de methode `loadString()`, geplaatst in de toepassingssandbox. (De standaardwaarde is `false`.) Inhoud die wordt geladen via de methode `loadString()`, krijgt daarmee toegang tot de eigenschap `window.runtime` en tot alle API's van AIR. Als u deze eigenschap instelt op `true`, moet u ervoor zorgen dat de gegevensbron voor een tekenreeks die wordt gebruikt in een aanroep naar de methode `loadString()`, vertrouwd wordt. Code-instructies in de HTML-tekenreeks worden uitgevoerd met alle toepassingsmachtigingen wanneer deze eigenschap is ingesteld op `true`. Stel deze eigenschap alleen in op `true` wanneer u er zeker van bent dat de tekenreeks geen schadelijke code kan bevatten.

In toepassingen die zijn gecompileerd met de SDK's van AIR 1.0 of AIR 1.1, wordt inhoud die wordt geladen via de methode `loadString()`, geplaatst in de toepassingssandbox.

Belangrijke veiligheidsregels bij gebruik van HTML in AIR-toepassingen

Adobe AIR 1.0 of hoger

De bestanden die u met de AIR-toepassing installeert, hebben toegang tot de API's van AIR. Om veiligheidsredenen heeft de inhoud van andere bronnen geen toegang tot deze API's. Voorbeeld: deze beperking voorkomt dat inhoud van een extern domein (bijvoorbeeld http://example.com) de inhoud van de bureaubladmap van de gebruiker (of nog erger) kan lezen.

Omdat beveiligingshiaten kunnen worden gebruikt door de functie `eval()` (en verwante API's) op te roepen, kan inhoud die met de toepassing wordt geïnstalleerd, deze methoden standaard niet gebruiken. Bepaalde Ajax-frameworks gebruiken echter de functie `eval()` en verwante API's.

Om inhoud goed te structureren zodat deze kan werken in een AIR-toepassing, moet u rekening houden met de beveiligingsbeperkingen voor inhoud van verschillende bronnen. Inhoud van verschillende locaties wordt in aparte beveiligingsclassificaties, sandboxen, geplaatst (zie "Beveiligingssandboxen" op pagina 1108). Inhoud die met de toepassing wordt geïnstalleerd, wordt standaard in een sandbox geïnstalleerd die de *toepassingssandbox* wordt genoemd. Deze verleent toegang tot de API's van AIR. De toepassingssandbox is de veiligste sandbox, aangezien beperkingen voorkomen dat niet-vertrouwde code wordt uitgevoerd.

U kunt inhoud die met uw toepassing is geïnstalleerd, ook in een andere sandbox plaatsen. Inhoud in niet-toepassingssandboxen werkt in een beveiligde omgeving, zoals in een webbrowser. Voorbeeld: code in niet-toepassingssandboxen kan `eval()` en verwante methoden gebruiken (maar anderzijds heeft deze code geen toegang tot de API's van AIR). De runtime voorziet in manieren om inhoud in verschillende sandboxen veilig te laten communiceren (zonder bijvoorbeeld API's van AIR toegankelijk te maken voor niet-toepassingsinhoud). Zie "Cross-scripting van inhoud in verschillende beveiligingssandboxen" op pagina 1058 voor meer informatie.

Als u code oproept die niet in een sandbox kan worden gebruikt vanwege beveiligingsbeperkingen, geeft de runtime de foutmelding weer dat er in de runtime van Adobe AIR een beveiligingsfout betreffende JavaScript-code in de toepassingsbeveiligingssandbox is opgetreden.

Om deze fout te voorkomen, volgt u de coderingsrichtlijnen in de volgende sectie, "JavaScript-beveiligingsfouten voorkomen" op pagina 1041.

Zie "HTML-beveiliging in Adobe AIR" op pagina 1149 voor meer informatie.

JavaScript-beveiligingsfouten voorkomen

Adobe AIR 1.0 of hoger

Als u code oproept die niet in een sandbox kan worden gebruikt vanwege deze beveiligingsbeperkingen, geeft de runtime de foutmelding weer dat er in de runtime van Adobe AIR een beveiligingsfout betreffende JavaScript-code in de toepassingsbeveiligingssandbox is opgetreden. Ga als volgt te werk om deze fout te voorkomen.

Oorzaken van JavaScript-beveiligingsfouten

Adobe AIR 1.0 of hoger

Code die wordt uitgevoerd in de toepassingssandbox, is uitgesloten van de meeste bewerkingen waarbij tekenreeksen worden geëvalueerd en uitgevoerd nadat de documentgebeurtenis `load` is geactiveerd en eventuele handlers voor de gebeurtenis `load` zijn afgesloten. Als u de volgende typen JavaScript-instructies gebruikt die potentieel onveilige tekenreeksen evalueren en uitvoeren, worden JavaScript-fouten gegenereerd:

- eval(), functie
- setTimeout() en setInterval()
- Functieconstructor

  Bovendien mislukken de volgende typen JavaScript-instructies zonder een Javascript-fout voor onveilige inhoud te genereren:

- javascript: URL's
- Gebeurteniscallbacks die via onevent-kenmerken in innerHTML- en outerHTML-instructies zijn toegewezen
- JavaScript-bestanden laden van buiten de installatiemap van de toepassing

- document.write() en document.writeln()
- Synchrone XMLHttpRequests vóór de gebeurtenis load of tijdens de gebeurtenishandler load
- Dynamisch gecreëerde scriptelementen

Opmerking: *In enkele beperkte gevallen is de evaluatie van tekenreeksen toegestaan. Zie "Codebeperkingen voor de inhoud van verschillende sandboxen" op pagina 1152 voor meer informatie.*

Adobe houdt een lijst bij van Ajax-frameworks die de toepassingsbeveiligingssandbox ondersteunen. Deze lijst vindt u op http://www.adobe.com/go/airappsandboxframeworks_nl.

In de volgende secties vindt u informatie om scripts zo aan te passen, dat deze JavaScript-fouten voor onveilige inhoud en stille defecten worden voorkomen voor code die in de toepassingssandbox wordt uitgevoerd.

Toepassingsinhoud toewijzen aan een andere sandbox

Adobe AIR 1.0 of hoger

In de meeste gevallen kunt u een toepassing aanpassen of opnieuw structureren, zodat JavaScript-beveiligingsfouten worden voorkomen. Als aanpassen of opnieuw structureren echter niet mogelijk is, kunt u de inhoud van de toepassing in een andere sandbox laden met de techniek die is beschreven in "Toepassingsinhoud laden in een niet-toepassingssandbox" op pagina 1059. Als deze inhoud ook toegang moet hebben tot de API's van AIR, kunt u een sandboxbridge maken, zoals beschreven in "Sandboxbridge-interface instellen" op pagina 1059.

eval(), functie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In de toepassingssandbox kunt u de functie `eval()` alleen gebruiken vóór de gebeurtenis `load` van de pagina of tijdens de gebeurtenishandler `load`. Nadat de pagina is geladen, wordt geen code uitgevoerd als `eval()` wordt opgeroepen. In de volgende gevallen kunt u uw code echter aanpassen om het gebruik van `eval()` te voorkomen.

Eigenschappen toewijzen aan een object

Adobe AIR 1.0 of hoger

In plaats van een tekenreeks te parseren om de eigenschapsaccessor te bouwen:

```
eval("obj." + propName + " = " + val);
```

roep eigenschappen op met haakjesnotering:

```
obj[propName] = val;
```

Functies maken met variabelen die beschikbaar zijn in de context

Adobe AIR 1.0 of hoger

Vervang instructies zoals deze:

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

door:

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

Objecten maken met de naam van de klasse als tekenreeksparameter

Adobe AIR 1.0 of hoger

Bekijken we een hypothetische JavaScript-klasse die met de volgende code is gedefinieerd:

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

De eenvoudigste manier om een instantie te maken, is met behulp van `eval()`:

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

U kunt het oproepen van `eval()` echter voorkomen door elke component van de klassennaam te parseren en het nieuwe object op te bouwen met behulp van de haakjesnotering:

```
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;
}
```

Om de instantie te maken, gebruikt u:

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() en setInterval()

Adobe AIR 1.0 of hoger

Vervang de tekenreeks die als handlerfunctie is opgegeven, door een functieverwijzing of een object. Voorbeeld: vervang een instructie zoals:

```
setTimeout("alert('Timeout')", 100);
```

door:

```
setTimeout(function(){alert('Timeout')}, 100);
```

Of, als de functie vereist dat het object `this` wordt ingesteld door de oproeper, vervangt u een instructie zoals:

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

door het volgende:

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

Functieconstructor

Adobe AIR 1.0 of hoger

Oproepen van `new Function(param, body)` kunnen worden vervangen door een inline functiedeclaratie of alleen worden gebruikt voordat de gebeurtenis `load` van de pagina is afgehandeld.

javascript: URL's

Adobe AIR 1.0 of hoger

De code die in een koppeling met behulp van het URL-schema javascript: is gedefinieerd, wordt in de toepassingssandbox genegeerd. Er wordt geen JavaScript-fout voor onveilige inhoud gegenereerd. U kunt koppelingen vervangen die gebruikmaken van URL's van het type javascript:, zoals:

```
<a href="javascript:code()">Click Me</a>
```

door:

```
<a href="#" onclick="code()">Click Me</a>
```

Gebeurteniscallbacks die via onevent-kenmerken in innerHTML- en outerHTML-instructies zijn toegewezen

Adobe AIR 1.0 of hoger

Als u innerHTML of outerHTML gebruikt om elementen toe te voegen aan de DOM van een document, worden gebeurteniscallbacks die in de instructie zijn toegewezen, zoals `onclick` of `onmouseover`, genegeerd. Er wordt geen beveiligingsfout gegenereerd. In plaats daarvan kunt u het kenmerk `id` aan de nieuwe elementen toewijzen en de callbackfuncties van de gebeurtenishandler instellen met de methode `addEventListener()`.

U hebt bijvoorbeeld een doelelement in een document, zoals:

```
<div id="container"></div>
```

Vervang instructies zoals:

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

door:

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

JavaScript-bestanden laden van buiten de installatiemap van de toepassing

Adobe AIR 1.0 of hoger

Scriptbestanden kunnen niet van buiten de toepassingssandbox worden geladen. Er wordt geen beveiligingsfout gegenereerd. Alle scriptbestanden die in de toepassingssandbox worden uitgevoerd, moeten in de toepassingsmap zijn geïnstalleerd. Als u externe scripts in een pagina wenst te gebruiken, moet u de pagina toewijzen aan een andere sandbox. Zie "Toepassingsinhoud laden in een niet-toepassingssandbox" op pagina 1059.

document.write() en document.writeln()

Adobe AIR 1.0 of hoger

Oproepen van `document.write()` of `document.writeln()` worden genegeerd nadat de gebeurtenis `load` van de pagina is afgehandeld. Er wordt geen beveiligingsfout gegenereerd. Als alternatief kunt u een nieuw bestand laden of de hoofdtekst van het document vervangen met behulp van DOM-manipulatietechnieken.

Synchrone XMLHttpRequests vóór de gebeurtenis load of tijdens de gebeurtenishandler load

Adobe AIR 1.0 of hoger

Synchrone XMLHttpRequests die worden gestart vóór de gebeurtenis `load` van de pagina of tijdens de handler voor de gebeurtenis `load`, retourneren geen inhoud. Asynchrone XMLHttpRequests kunnen worden gestart maar retourneren pas na de gebeurtenis `load`. Nadat de gebeurtenis `load` is afgehandeld, gedragen synchrone XMLHttpRequests zich op de normale manier.

Dynamisch gecreëerde scriptelementen

Adobe AIR 1.0 of hoger

Scriptelementen die dynamisch zijn gecreëerd, bijvoorbeeld met innerHTML of de methode `document.createElement()`, worden genegeerd.

API-klassen van AIR oproepen vanuit JavaScript

Adobe AIR 1.0 of hoger

Naast de standaardelementen en de uitgebreide elementen van Webkit, heeft HTML- en JavaScript-code toegang tot de hostklassen die door de runtime worden geboden. Met deze klassen hebt u toegang tot de geavanceerde functies van AIR, zoals:

- Toegang tot het bestandssysteem
- Gebruik van lokale SQL-databases
- Besturing van toepassings- en venstermenu's
- Toegang tot netwerksockets
- Gebruik van door de gebruiker gedefinieerde klassen en objecten
- Geluidsmogelijkheden

  Voorbeeld: de AIR API voor het bestandssysteem bevat de klasse File, die in het pakket flash.filesystem is opgenomen. U kunt als volgt een File-object in JavaScript maken:

  ```
  var myFile = new window.runtime.flash.filesystem.File();
  ```

  Het `runtime`-object is een speciaal JavaScript-object, dat beschikbaar is voor HTML-inhoud die in AIR in de toepassingssandbox wordt uitgevoerd. Hiermee hebt u vanuit JavaScript toegang tot runtimeklassen. De eigenschap `flash` van het `runtime`-object biedt toegang tot het pakket flash. De eigenschap `flash.filesystem` van het `runtime`-object biedt toegang tot het pakket flash.filesystem (dit pakket bevat de klasse File). Pakketten zijn een manier om klassen te organiseren die in ActionScript worden gebruikt.

  Opmerking: *De eigenschap `runtime` wordt niet automatisch toegevoegd aan de vensterobjecten van pagina's die in een frame of iframe zijn geladen. Zolang het onderliggende document zich echter in de toepassingssandbox bevindt, heeft het onderliggende element toegang tot de eigenschap `runtime` van het bovenliggende element.*

  Vanwege de pakketstructuur van de runtimeklassen zou een ontwikkelaar lange reeksen JavaScript-code moeten typen om toegang te krijgen tot elke klasse (bijvoorbeeld `window.runtime.flash.desktop.NativeApplication`). Daarom omvat de SDK van AIR het bestand AIRAliases.js, waarmee u veel makkelijker toegang hebt tot runtimeklassen (door bijvoorbeeld gewoon `air.NativeApplication` te typen).

  De API-klassen van AIR worden in deze handleiding besproken. Andere klassen van de API Flash Player, die interessant kunnen zijn voor HTML-ontwikkelaars, worden beschreven in de *Adobe AIR API-naslaggids voor HTML-ontwikkelaars*. ActionScript is de taal die wordt gebruikt in SWF-inhoud (Flash Player). De syntaxis van JavaScript en die van ActionScript zijn echter bijna identiek. (Ze zijn allebei gebaseerd op versies van de taal ECMAScript.) Alle ingebouwde klassen zijn zowel in JavaScript (in HTML-inhoud) als in ActionScript (in SWF-inhoud) beschikbaar.

  Opmerking: *JavaScript-code kan geen gebruik maken van de klassen Dictionary, XML en XMLList, die wel beschikbaar zijn in ActionScript.*

Het bestand AIRAliases.js gebruiken

Adobe AIR 1.0 of hoger

De runtimeklassen zijn georganiseerd in een pakketstructuur, zoals hierna wordt geïllustreerd:

- `window.runtime.flash.desktop.NativeApplication`
- `window.runtime.flash.desktop.ClipboardManager`
- `window.runtime.flash.filesystem.FileStream`
- `window.runtime.flash.data.SQLDatabase`

In de SDK van AIR is ook het bestand AIRAliases.js opgenomen, dat "alias"-definities bevat waarmee u runtimeklassen kunt oproepen zonder omslachtig typewerk. Voorbeeld: u kunt de klassen die hierboven zijn vermeld, oproepen door gewoon het volgende te typen:

- `air.NativeApplication`
- `air.Clipboard`
- `air.FileStream`
- `air.SQLDatabase`

Deze lijst is slechts een kort fragment van de klassen in het bestand AIRAliases.js. De volledige lijst klassen en functies op pakketniveau vind u in de *Adobe AIR API-naslaggids voor HTML-ontwikkelaars*.

Naast de vaak gebruikte runtimeklassen bevat het bestand AIRAliases.js ook aliassen voor vaak gebruikte functies op pakketniveau: `window.runtime.trace()`, `window.runtime.flash.net.navigateToURL()` en `window.runtime.flash.net.sendToURL()`, met als alias `air.trace()`, `air.navigateToURL()` en `air.sendToURL()`.

Als u het bestand AIRAliases.js wilt gebruiken, neemt u de volgende `script`-verwijzing op in uw HTML-pagina:

```
<script src="AIRAliases.js"></script>
```

Pas indien nodig het pad aan in de `src`-verwijzing.

Belangrijk: *Tenzij anders is aangegeven, gaat de JavaScript-voorbeeldcode in deze documentatie ervan uit dat u het bestand AIRAliases.js hebt opgenomen in uw HTML-pagina.*

Informatie over URL's in AIR

Adobe AIR 1.0 of hoger

In HTML-inhoud die in AIR wordt uitgevoerd, kunt u een van de volgende URL-schema's gebruiken om `src`-kenmerken te definiëren voor de tags `img`, `frame`, `iframe` en `script`, in het kenmerk `href` van de tag `link`, of op een willekeurige plaats waar u een URL kunt opgeven.

URL-schema	Beschrijving	Voorbeeld
file	Het pad op basis van de hoofdmap van het bestandssysteem.	`file:///c:/AIR Test/test.txt`
app	Het pad op basis van de hoofdmap van de geïnstalleerde toepassing.	`app:/images`
app-storage	Het pad op basis van de opslagmap van de toepassing. Voor elke geïnstalleerde toepassing definieert AIR een unieke opslagmap. Dit is de plaats waar gegevens worden opgeslagen die eigen zijn aan die toepassing.	`app-storage:/settings/prefs.xml`
http	Een standaard HTTP-aanvraag.	`http://www.adobe.com`
https	Een standaard HTTPS-aanvraag.	`https://secure.example.com`

Zie “URI-schema's” op pagina 860 voor meer informatie over het gebruik van URL-schema's in AIR.

API's van AIR, zoals de klassen File, Loader, URLStream en Sound, maken vaak gebruik van een URLRequest-object in plaats van een tekenreeks waarin de URL is opgenomen. Het URLRequest-object zelf wordt geïnitialiseerd met een tekenreeks, die dezelfde URL-schema's kan gebruiken. Voorbeeld: de volgende instructie creëert een URLRequest-object waarmee de startpagina van Adobe kan worden opgevraagd:

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

Zie “HTTP-communicatie” op pagina 858 voor meer informatie over URLRequest-objecten.

ActionScript-objecten beschikbaar maken voor JavaScript

Adobe AIR 1.0 of hoger

JavaScript in de HTML-pagina die door een HTMLLoader-object is geladen, kan de klassen, objecten en functies oproepen die in de ActionScript-uitvoeringscontext zijn gedefinieerd. Daartoe wordt gebruikgemaakt van de eigenschappen `window.runtime`, `window.htmlLoader` en `window.nativeWindow` van de HTML-pagina. U kunt ook ActionScript-objecten en -functies beschikbaar maken voor JavaScript-code door verwijzingen naar die objecten en functies op te nemen in de JavaScript-uitvoeringscontext.

Basisvoorbeeld om JavaScript-objecten op te roepen vanuit ActionScript

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u eigenschappen kunt toevoegen die ActionScript-objecten verwijzen naar het algemene vensterobject van een HTML-pagina:

```
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;
}
```

De HTML-inhoud (in een bestand met de naam test.html), die in het vorige voorbeeld in het HTMLLoader-object is geladen, kan de eigenschap `foo` oproepen, evenals de methode `helloFromJS()`, die in het bovenliggende SWF-bestand is gedefinieerd:

```
<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>
```

Als u de JavaScript-context oproept van een document dat op dat moment wordt geladen, kunt u met de gebeurtenis `htmlDOMInitialize` al vroeg in de constructiesequentie van de pagina objecten maken, zodat ze toegankelijk zijn voor scripts die in de pagina zijn gedefinieerd. Als u wacht op de gebeurtenis `complete`, kunnen alleen scripts die na de gebeurtenis `load` van de pagina worden uitgevoerd, toegang krijgen tot de toegevoegde objecten.

Klassedefinities beschikbaar maken voor JavaScript

Adobe AIR 1.0 of hoger

Als u de ActionScript-klassen van uw toepassing beschikbaar wilt maken in JavaScript, kunt u de geladen HTML-inhoud toewijzen aan het toepassingsdomein waarin zich de klassedefinities bevinden. Het toepassingsdomein van de JavaScript-uitvoeringscontext kan worden ingesteld met de eigenschap `runtimeApplicationDomain` van het HTMLLoader-object. Als u het toepassingsdomein bijvoorbeeld wilt instellen op het primaire toepassingsdomein, stelt u `runtimeApplicationDomain` in op `ApplicationDomain.currentDomain`, zoals in de volgende code:

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

Nadat de eigenschap `runtimeApplicationDomain` is ingesteld, deelt de JavaScript-context klassedefinities met het toegewezen domein. Als u een instantie van een aangepaste klasse in JavaScript wilt maken, verwijst u naar de klassedefinitie via de eigenschap `window.runtime` en gebruikt u de operator `new`:

```
var customClassObject = new window.runtime.CustomClass();
```

De HTML-inhoud moet afkomstig zijn van een compatibel beveiligingsdomein. Als de HTML-inhoud afkomstig is van een ander beveiligingsdomein dan dat van het toepassingsdomein dat u toewijst, gebruikt de pagina in plaats daarvan een standaard toepassingsdomein. Voorbeeld: als u een externe pagina van internet laadt, kunt u ApplicationDomain.currentDomain niet toewijzen als het toepassingsdomein van de pagina.

Gebeurtenislisteners verwijderen

Adobe AIR 1.0 of hoger

Als u gebeurtenislisteners voor JavaScript toevoegt aan objecten buiten de huidige pagina, inclusief runtimeobjecten, objecten in de geladen SWF-inhoud en zelfs JavaScript-objecten die in andere pagina's worden uitgevoerd, moet u deze gebeurtenislisteners altijd verwijderen als de pagina niet meer is geladen. Anders verzendt de gebeurtenislistener de gebeurtenis naar een handlerfunctie die niet meer bestaat. In dit geval wordt er een foutmelding weergegeven dat de toepassing heeft geprobeerd te verwijzen naar een JavaScript-object in een HTML-pagina die niet meer is geladen. Wanneer u overbodige gebeurtenislisteners verwijdert, kan AIR ook het overeenkomstige geheugen opnieuw gebruiken. Zie "Gebeurtenislisteners verwijderen in HTML-pagina's die navigeren" op pagina 1087 voor meer informatie.

HTML DOM- en JavaScript-objecten oproepen vanuit ActionScript

Adobe AIR 1.0 of hoger

Nadat het HTMLLoader-object de gebeurtenis `complete` heeft verzonden, hebt u toegang tot alle objecten in het HTML DOM (Document Object Model) voor de pagina. Toegankelijke objecten zijn onder andere weergave-elementen (zoals de objecten `div` en `p` in de pagina), evenals JavaScript-variabelen en -functies. De gebeurtenis `complete` komt overeen met de JavaScript-gebeurtenis `load` van de pagina. Vóór `complete` wordt verzonden, mogen variabelen, functies en DOM-elementen niet zijn geparseerd of gecreëerd. Indien mogelijk wacht u op de gebeurtenis `complete` vóór u het HTML DOM benadert.

Bekijken we als voorbeeld de volgende HTML-pagina:

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

Deze eenvoudige HTML-pagina definieert een JavaScript-variabele met de naam *foo* en een JavaScript-functie met de naam *test()*. Beide zijn eigenschappen van het algemene object `window` van de pagina. Bovendien bevat het object `window.document` een element met de naam P (met de id *p1*), dat u kunt oproepen met de methode `getElementById()`. Nadat de pagina is geladen (wanneer het HTMLLoader-object de gebeurtenis `complete` verzendt), kunt u al deze objecten benaderen vanuit ActionScript, zoals wordt geïllustreerd met de volgende ActionScript-code:

```
 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.
}
```

Als u de inhoud van een HTML-element wilt oproepen, gebruikt u de eigenschap `innerHTML`. De vorige code gebruikt bijvoorbeeld `html.window.document.getElementById("p1").innerHTML` om de inhoud van het HTML-element *p1* op te halen.

U kunt ook eigenschappen voor de HTML-pagina instellen vanuit ActionScript. Het volgende voorbeeld stelt de inhoud van het element `p1` en de waarde van de JavaScript-variabele `foo` op de pagina in met behulp van een verwijzing naar het bevattende HTMLLoader-object:

```
 html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

SWF-inhoud insluiten in HTML

Adobe AIR 1.0 of hoger

In een AIR-toepassing kunt u SWF-inhoud insluiten in HTML-inhoud, net zoals in een browser. Sluit de SWF-inhoud in met behulp van de tag `object`, de tag `embed` of beide.

Opmerking: *Vaak wordt zowel de tag `object` als de tag `embed` gebruikt om SWF-inhoud weer te geven in een HTML-pagina. Deze techniek biedt echter geen voordelen in AIR. U kunt de W3C-standaard tag `object` op zichzelf gebruiken in inhoud die in AIR moet worden weergegeven. U kunt ook desgewenst de tags `object` en `embed` samen blijven gebruiken voor HTML-inhoud die ook in een browser wordt weergegeven.*

Als u transparantie hebt ingeschakeld in het NativeWindow-object waarin de HTML- en SWF-inhoud wordt weergegeven, geeft AIR de SWF-inhoud niet weer wanneer de voor het insluiten van de inhoud gebruikte venstermodus (`wmode`) is ingesteld op de waarde `window`. Als u SWF-inhoud wilt weergeven in een HTML-pagina van een transparant venster, stelt u de parameter `wmode` in op `opaque` of `transparent`. `window` is de standaardwaarde voor `wmode`. Als u dus geen waarde opgeeft, is het mogelijk dat uw inhoud niet wordt weergegeven.

In het volgende voorbeeld ziet u hoe u de HTML-tag `object` gebruikt om een SWF-bestand in HTML-inhoud weer te geven. De parameter `wmode` is ingesteld op `opaque`, zodat de inhoud wordt weergegeven, ook als het onderliggende NativeWindow-object transparant is. Het SWF-bestand wordt vanuit de toepassingsmap geladen maar u kunt een willekeurig URL-schema gebruiken dat door AIR wordt ondersteund. (De bronlocatie van het SWF-bestand bepaalt de beveiligingssandbox waarin AIR de inhoud plaatst.)

```
<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>
```

U kunt ook een script gebruiken om inhoud dynamisch te laden. In het volgende voorbeeld ziet u hoe u een `object`-knooppunt maakt om het SWF-bestand weer te geven dat is opgegeven in de parameter `urlString`. Het voorbeeld voegt het knooppunt toe als onderliggend element van het pagina-element met de id die is opgegeven met de parameter `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>
```

SWF-inhoud wordt niet weergegeven als het HTMLLoader-object geschaald of geroteerd is of als de eigenschap `alpha` is ingesteld op een andere waarde dan 1.0. Vóór AIR 1.5.2 werd SWF-inhoud niet weergegeven in een transparant venster, ongeacht de voor `wmode` ingestelde waarde.

Opmerking: *Wanneer een ingesloten SWF-object probeert een extern element te laden, zoals een videobestand, is het mogelijk dat de SWF-inhoud niet op de juiste wijze wordt gerenderd wanneer geen absoluut pad naar het videobestand wordt aangegeven in het HTML-bestand. Een ingesloten SWF-object kan echter wel een extern afbeeldingsbestand laden met gebruik van een relatief pad.*

Aan de hand van het volgende voorbeeld wordt geïllustreerd hoe externe elementen kunnen worden geladen via een SWF-object dat is ingesloten in HTML-inhoud:

```
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");

}
```

In het volgende ActionScript-voorbeeld wordt het door het HTML-bestand doorgegeven afbeeldingspad gelezen en wordt de afbeelding geladen in het werkgebied:

```
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;
                  }
            }
      }
}
```

ActionScript-bibliotheken in een HTML-pagina gebruiken

Adobe AIR 1.0 of hoger

AIR breidt het HTML-scriptelement uit, zodat een pagina ActionScript-klassen kan importeren in een gecompileerd SWF-bestand. Stel dat de bibliotheek *myClasses.swf* zich in de submap `lib` van de hoofdtoepassingsmap bevindt. Als u deze bibliotheek wilt importeren, neemt u de volgende scripttag op in een HTML-bestand:

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

Belangrijk: *Het kenmerk type moet* `type="application/x-shockwave-flash"` *zijn, anders wordt de bibliotheek niet goed geladen.*

Als de SWF-inhoud is gecompileerd als Flash Player 10 of AIR 1.5 SWF, moet u de XML-naamruimte van het descriptorbestand van de toepassing instellen op de AIR 1.5-naamruimte.

De map `lib` en het bestand `myClasses.swf` moeten ook worden opgenomen wanneer het AIR-bestand in een pakket wordt geplaatst.

Benader de geïmporteerde klassen via de eigenschap `runtime` van het JavaScript-object Window:

```
var libraryObject = new window.runtime.LibraryClass();
```

Als de klassen in het SWF-bestand zijn verdeeld in pakketten, moet u ook de pakketnaam opnemen. Als de definitie LibraryClass zich bijvoorbeeld in een pakket met de naam *utilities* bevindt, maakt u met de volgende instructie een instantie van de klasse:

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

Opmerking: *Als u in AIR een ActionScript SWF-bibliotheek wilt compileren als onderdeel van een HTML-pagina, gebruikt u de compiler* `acompc`*. Het hulpprogramma acompc maakt deel uit van de SDK van Flex en wordt beschreven in de Flex SDK-documentatie.*

HTML DOM- en JavaScript-objecten oproepen vanuit een geïmporteerd ActionScript-bestand

Adobe AIR 1.0 of hoger

Als u objecten in een HTML-pagina wenst op te roepen vanuit ActionScript in een SWF-bestand dat in de pagina is geïmporteerd met behulp van de tag `<script>`, geeft u een verwijzing naar een JavaScript-object zoals `window` of `document` op aan een functie die in de ActionScript-code is gedefinieerd. Gebruik de verwijzing binnen de functie om het JavaScript-object op te roepen (of andere objecten die toegankelijk zijn via de doorgegeven verwijzing).

Bekijken we als voorbeeld de volgende HTML-pagina:

```
<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>
```

Deze eenvoudige HTML-pagina heeft een JavaScript-variabele met de naam *num* en een JavaScript-functie met de naam *getStatus()*. Beide zijn eigenschappen van het object `window` van de pagina. Bovendien bevat het object `window.document` een element met de naam P (met de id *p1*).

De pagina laadt het ActionScript-bestand "ASLibrary.swf", waarin zich de klasse ASClass bevindt. ASClass definieert de functie `accessDOM()`, die de waarden van deze JavaScript-objecten bijhoudt. De methode `accessDOM()` gebruikt het JavaScript-object Window als argument. Op basis van deze Window-verwijzing kunnen andere objecten in de pagina worden benaderd, zoals variabelen, functies en DOM-elementen, zoals wordt geïllustreerd in de volgende definitie:

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

U kunt eigenschappen van de HTML-pagina ophalen en instellen op basis van een geïmporteerde ActionScript-klasse. De volgende functie stelt bijvoorbeeld de inhoud van het element `p1` op de pagina in, evenals de waarde van de JavaScript-variabele `foo` op de pagina:

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Date- en RegExp-objecten converteren

Adobe AIR 1.0 of hoger

De talen JavaScript en ActionScript definiëren allebei de klassen Date en RegExp, maar objecten van deze typen worden niet automatisch geconverteerd tussen de twee uitvoeringscontexten. U moet de Date- en RegExp-objecten naar het equivalente type converteren vóór u ze gebruikt om eigenschappen of functieparameters in de andere uitvoeringscontext in te stellen.

Voorbeeld: de volgende ActionScript-code converteert het JavaScript-object Date met de naam `jsDate` naar het ActionScript-object Date:

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

De volgende ActionScript-code converteert het JavaScript-object RegExp met de naam `jsRegExp` naar het ActionScript-object RegExp:

```
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);
```

HTML-opmaakmodellen bewerken vanuit ActionScript

Adobe AIR 1.0 of hoger

Nadat het HTMLLoader-object de gebeurtenis `complete` heeft verzonden, kunt u CSS-stijlen in een pagina onderzoeken en bewerken.

Bekijken we als voorbeeld het volgende eenvoudige HTML-document:

```
<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>
```

Nadat een HTMLLoader-object deze inhoud heeft geladen, kunt u de CSS-stijlen in de pagina als volgt bewerken via de array `cssRules` van de array `window.document.styleSheets`:

```
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";
}
```

Deze code past de CSS-stijlen zo aan, dat het resulterende HTML-document er als volgt uitziet:

Style 1A

Style 1B

Style 2

Let op: code kan stijlen aan de pagina toevoegen nadat het HTMLLoader-object de gebeurtenis `complete` heeft verzonden.

Cross-scripting van inhoud in verschillende beveiligingssandboxen

Adobe AIR 1.0 of hoger

Het beveiligingsmodel van de runtime isoleert code op basis van de oorsprong. Door cross-scripting van de inhoud in verschillende beveiligingssandboxen kunt u inhoud in een bepaalde beveiligingssandbox toegang bieden tot bepaalde eigenschappen en methoden in een andere sandbox.

AIR-beveiligingssandboxen en JavaScript-code

Adobe AIR 1.0 of hoger

AIR past een zelfde-oorsprongbeleid toe, dat voorkomt dat code in een bepaald domein kan interageren met inhoud in een ander domein. Alle bestanden worden in een sandbox geplaatst op basis van hun oorsprong. Inhoud in de toepassingssandbox kan normaal gesproken niet het zelfde-oorsprongprincipe schenden en inhoud die van buiten de installatiemap van de toepassing is geladen, cross-scripten. AIR biedt echter een aantal technieken waarmee u cross-scripting kunt toepassen op niet-toepassingsinhoud.

Eén techniek maakt gebruik van frames of iframes om toepassingsinhoud aan een andere beveiligingssandbox toe te wijzen. Pagina's die vanaf de sandboxzone van de toepassing zijn geladen, gedragen zich alsof ze vanaf het externe domein zijn geladen. Voorbeeld: als u toepassingsinhoud toewijst aan het domein *example.com*, kan die inhoud pagina's cross-scripten die vanaf example.com zijn geladen.

Deze techniek plaatst de toepassingsinhoud in een andere sandbox, zodat code binnen die inhoud niet langer is onderworpen aan de uitvoeringsbeperkingen voor code in geëvalueerde tekenreeksen. U kunt deze techniek van sandboxtoewijzing gebruiken om deze beperkingen te verminderen, zelfs als u geen cross-scripting van externe inhoud hoeft uit te voeren. Inhoud op die manier toewijzen, kan met name interessant zijn als u met een van de vele JavaScript-frameworks werkt, of met bestaande code die is gebaseerd op de evaluatie van tekenreeksen. Neem echter het extra risico in acht dat onvertrouwde inhoud kan worden geïnjecteerd en uitgevoerd als inhoud buiten de toepassingssandbox wordt uitgevoerd. Pas hiervoor de nodige voorzorgsmaatregelen toe.

Anderzijds verliest toepassingsinhoud die aan een andere sandbox is toegewezen, de toegangsrechten voor de API's van AIR, zodat de techniek van sandboxtoewijzing niet kan worden gebruikt om AIR-functionaliteit toegankelijk te maken voor code die buiten de toepassingssandbox wordt uitgevoerd.

Met een andere cross-scriptingtechniek kunt u een interface (*sandboxbridge*) maken tussen inhoud in een niet-toepassingssandbox en het bovenliggende document in de toepassingssandbox. Dankzij deze bridge heeft de onderliggende inhoud toegang tot eigenschappen en methoden die door de bovenliggende inhoud worden gedefinieerd, en omgekeerd. Beide situaties kunnen zich ook tegelijk voordoen.

Ten slotte kunt u ook domeinoverschrijdende XMLHttpRequests uitvoeren vanuit de toepassingssandbox en optioneel vanuit andere sandboxen.

Zie “Frame- en iframe-elementen in HTML” op pagina 1030, “HTML-beveiliging in Adobe AIR” op pagina 1149 en “XMLHttpRequest-objecten” op pagina 1024 voor meer informatie.

Toepassingsinhoud laden in een niet-toepassingssandbox

Adobe AIR 1.0 of hoger

Om toepassingsinhoud veilig te cross-scripten met inhoud die van buiten de installatiemap van de toepassing is geladen, kunt u het element `frame` of `iframe` gebruiken om toepassingsinhoud in dezelfde beveiligingssandbox als de externe inhoud te laden. Als u geen cross-scripting van de externe inhoud nodig hebt maar toch een pagina van uw toepassing buiten de toepassingssandbox wilt laden, kunt u dezelfde techniek gebruiken, waarbij u `http://localhost/` of een andere onschuldige waarde opgeeft als het domein of de oorsprong.

AIR voegt de nieuwe kenmerken `sandboxRoot` en `documentRoot` toe aan het element frame, zodat u kunt bepalen of een toepassingsbestand dat in het frame is geladen, moet worden toegewezen aan een niet-toepassingssandbox. Bestanden met een pad onder de URL `sandboxRoot` worden in plaats daarvan geladen vanuit de map `documentRoot`. Om veiligheidsredenen wordt de toepassingsinhoud die op die manier wordt geladen, behandeld alsof deze vanaf de URL `sandboxRoot` is geladen.

De eigenschap `sandboxRoot` geeft de URL op die moet worden gebruikt om de sandbox en het domein te bepalen waarin de frame-inhoud moet worden geplaatst. Het URL-schema `file:`, `http:` of `https:` moet worden gebruikt. Als u een relatieve URL opgeeft, blijft de inhoud in de toepassingssandbox.

De eigenschap `documentRoot` bepaalt de map waaruit de frame-inhoud moet worden geladen. Het URL-schema `file:`, `app:` of `app-storage:` moet worden gebruikt.

In het volgende voorbeeld ziet u hoe u inhoud die in de submap `sandbox` van de toepassing is geïnstalleerd, toewijst voor uitvoering in de externe sandbox en het domein `www.example.com`:

```
 <iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

De pagina `ui.html` kan met behulp van de volgende scripttag een JavaScript-bestand laden vanuit de lokale map `sandbox`:

```
<script src="http://www.example.com/local/ui.js"></script>
```

Deze pagina kan ook inhoud laden vanuit een map op de externe server met bijvoorbeeld de volgende scripttag:

```
<script src="http://www.example.com/remote/remote.js"></script>
```

De URL `sandboxRoot` maskeert alle inhoud op dezelfde URL op de externe server. In het vorige voorbeeld hebt u geen toegang tot externe inhoud op `www.example.com/local/` (of een van de submappen) omdat AIR de aanvraag omleidt naar de lokale toepassingsmap. Alle aanvragen worden omgeleid, ongeacht of ze afkomstig zijn van paginanavigatie, een XMLHttpRequest of een andere manier om inhoud te laden.

Sandboxbridge-interface instellen

Adobe AIR 1.0 of hoger

U kunt een sandboxbridge gebruiken als inhoud in de toepassingssandbox toegang moet hebben tot eigenschappen of methoden die zijn gedefinieerd door inhoud in een niet-toepassingssandbox, of als niet-toepassingsinhoud toegang moet hebben tot eigenschappen en methoden die zijn gedefinieerd door inhoud in de toepassingssandbox. Creëer een bridge met de eigenschappen `childSandboxBridge` en `parentSandboxBridge` van het object `window` van een willekeurig onderliggend document.

Onderliggende sandboxbridges creëren

Adobe AIR 1.0 of hoger

Met de eigenschap `childSandboxBridge` kan het onderliggende document een interface toegankelijk kan maken voor inhoud in het bovenliggende document. Als u een interface toegankelijk wilt maken, moet u de eigenschap `childSandbox` instellen op een functie of object in het onderliggende document. Vervolgens hebt u toegang tot het object of de functie vanuit de inhoud van het bovenliggende document. In het volgende voorbeeld ziet u hoe een script dat in een onderliggend document wordt uitgevoerd, een object dat een functie en eigenschap bevat, toegankelijk kan maken voor het bovenliggende document:

```
var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Als deze onderliggende inhoud wordt geladen in een iframe met de id "child", kunt u de interface vanuit de bovenliggende inhoud benaderen door de eigenschap `childSandboxBridge` van het frame te lezen:

```
var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

Bovenliggende sandboxbridges creëren

Adobe AIR 1.0 of hoger

Met de eigenschap `parentSandboxBridge` kan het bovenliggende document een interface toegankelijk kan maken voor inhoud in een onderliggend document. Als u een interface toegankelijk wilt maken, stelt het bovenliggende document de eigenschap `parentSandbox` van een onderliggend document in op een functie of object dat in het bovenliggende document wordt gedefinieerd. Vervolgens hebt u toegang tot het object of de functie vanuit de inhoud van het onderliggende document. In het volgende voorbeeld ziet u hoe een script dat in een bovenliggend document wordt uitgevoerd, een object dat een functie bevat, toegankelijk kan maken voor een onderliggend document:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

Via deze interface kan inhoud in het onderliggende frame tekst opslaan in een bestand met de naam `save.txt`. Er is echter geen andere toegang tot het bestandssysteem. De onderliggende inhoud kan bijvoorbeeld de functie Save als volgt aanroepen:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

De toepassingsinhoud moet de kleinst mogelijke interface toegankelijk maken voor andere sandboxen. Niet-toepassingsinhoud moet standaard als onbetrouwbaar worden beschouwd omdat deze inhoud kan blootstaan aan toevallige of kwaadwillige injectie van code. U moet de nodige veiligheidsmaatregelen nemen om misbruik te voorkomen van de interface die u via de bovenliggende sandboxbridge toegankelijk maakt.

Bovenliggende sandboxbridges benaderen tijdens het laden van een pagina

Adobe AIR 1.0 of hoger

Als een script in een onderliggend document toegang moet hebben tot een bovenliggende sandboxbridge, moet de bridge worden ingesteld voordat het script wordt uitgevoerd. Window-, frame- en iframe-objecten verzenden de gebeurtenis `dominitialize` nadat een nieuw pagina-DOM is gemaakt maar voordat scripts zijn geparseerd of DOM-elementen zijn toegevoegd. U kunt de gebeurtenis `dominitialize` gebruiken om de bridge al vroeg in de constructiesequentie van de pagina te creëren, zodat alle scripts in het onderliggende document toegang hebben tot de bridge.

In het volgende voorbeeld ziet u hoe u een bovenliggende sandboxbridge kunt maken als reactie op de gebeurtenis `dominitialize` die vanuit het onderliggende frame is verzonden:

```
<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>
```

Het volgende document, `child.html`, illustreert hoe onderliggende inhoud de bovenliggende sandboxbridge kan benaderen:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Om zeker te zijn dat u naar de gebeurtenis `dominitialize` in een onderliggend venster luistert en niet naar een frame, moet u de listener aan het nieuwe onderliggende Window-object toevoegen, dat u creëert met de functie `window.open()`:

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

In dit geval bestaat er geen mogelijkheid om toepassingsinhoud toe te wijzen aan een niet-toepassingssandbox. Deze techniek is alleen nuttig als `child.html` van buiten de toepassingsmap wordt geladen. U kunt toepassingsinhoud in het venster nog altijd aan een niet-toepassingssandbox toewijzen, maar in dat geval moet u eerst een tussenpagina laden die zelf frames gebruikt om het onderliggende document te laden, en deze pagina toewijzen aan de gewenste sandbox.

Als u de functie `createRootWindow()` van de klasse HTMLLoader gebruikt om een venster te maken, is het nieuwe venster geen onderliggend element van het document waaruit `createRootWindow()` wordt opgeroepen. U kunt dus geen sandboxbridge maken tussen het oproepende venster en niet-toepassingsinhoud die in het nieuwe venster is geladen. In plaats daarvan moet u in het nieuwe venster een tussenpagina laden die zelf frames gebruikt om het onderliggende document te laden. U kunt vervolgens de bridge creëren tussen het bovenliggende document van het nieuwe venster en het onderliggende document dat in het frame is geladen.

Hoofdstuk 60: Scripting van de AIR HTML-container

Adobe AIR 1.0 of hoger

De klasse HTMLLoader fungeert als container voor HTML-inhoud in Adobe® AIR®. De klasse biedt talrijke eigenschappen en methoden, die zijn overgenomen van de klasse Sprite en waarmee u het gedrag en de weergave van het object in de ActionScript® 3.0-weergavelijst kunt bepalen. Daarnaast definieert de klasse eigenschappen en methoden voor taken zoals het laden van en communiceren met HTML-inhoud en het beheren van het historisch overzicht.

Met de klasse HTMLHost wordt een set met standaardgedragingen gedefinieerd voor een HTMLLoader. Wanneer u een HTMLLoader-object maakt, wordt niet voorzien in HTMLHost-implementatie. Dit betekent dat er niets gebeurt wanneer HTML-inhoud een van de standaardgedragingen activeert, zoals het wijzigen van de vensterlocatie of -titel. U kunt de klasse HTMLHost zo uitbreiden dat deze de gewenste gedragingen voor uw toepassing definieert.

Voor HTML-vensters die met AIR worden gemaakt, is voorzien in een standaardimplementatie van de HTMLHost. U kunt deze standaard HTMLHost-implementatie aan een ander HTMLLoader-object toewijzen door de eigenschap `htmlHost` van het object in te stellen met een nieuw HTMLHost-object dat is gemaakt met de parameter `defaultBehavior` op `true`.

Opmerking: *In het Adobe® Flex™ Framework wordt het HTMLLoader-object omwikkeld door de mx:HTML-component. Gebruik bij Flex de HTML-component.*

Eigenschappen van HTMLLoader-objecten weergeven

Adobe AIR 1.0 of hoger

Een HTMLLoader-object neemt de weergave-eigenschappen van de Adobe® Flash® Player-klasse Sprite. U kunt bijvoorbeeld vergroten, verkleinen, verplaatsen, verbergen en een andere achtergrondkleur selecteren. U kunt ook geavanceerde effecten toepassen, zoals filters, maskers, schaalinstelling en roteren. Houd bij het toepassen van effecten rekening met invloed op de leesbaarheid. SWF- en PDF-inhoud in een HTML-pagina kan niet worden weergegeven bij het toepassen van bepaalde effecten.

HTML-vensters bevatten een HTMLLoader-object dat de HTML-inhoud rendert. Aangezien dit object beperkt is tot het venstergebied, heeft het wijzigen van de afmetingen of positie, roteren of schalen niet altijd het gewenste resultaat.

Basisweergave-eigenschappen

Adobe AIR 1.0 of hoger

Met de basisweergave-eigenschappen van de HTMLLoader kunt u het besturingselement binnen het hoofdweergaveobject positioneren, de afmetingen instellen en weergeven/verbergen. U wordt aangeraden deze eigenschappen niet te wijzigen voor het HTMLLoader-object van een HTML-venster.

Dit zijn de basiseigenschappen:

Eigenschap	Opmerkingen
`x`, `y`	Positioneert het object binnen zijn hoofdcontainer.
`width`, `height`	Wijzigt de afmetingen van het weergavegebied.
`visible`	Regelt de zichtbaarheid van het object en eventuele inhoud ervan.

Aan de buitenkant van een HTML-venster zijn de eigenschappen `width` en `height` van een HTMLLoader-object standaard ingesteld op 0. U moet de breedte en hoogte instellen om de geladen HTML-inhoud zichtbaar te maken. HTML-inhoud is gekoppeld aan de HTMLLoader-afmetingen. De opmaak is afhankelijk van de HTML- en CSS-eigenschappen van de inhoud. Wanneer u de HTMLLoader-afmetingen wijzigt, wordt de loop van de inhoud aangepast.

Bij het laden van inhoud in een nieuw HTMLLoader-object (waarbij `width` nog altijd op 0 staat), kan het verleidelijk zijn om de waarden `width` en `height` voor de weergave van de HTMLLoader in te stellen met behulp van de eigenschappen `contentWidth` en `contentHeight`. Deze techniek werkt bij pagina's met een redelijke minimumbreedte die worden opgemaakt volgens de HTML- en CSS-regels voor de tekstloop. Sommige pagina's hebben echter een lange, smalle loop omdat er geen redelijke breedte is opgegeven door de HTMLLoader.

Opmerking: *Wanneer u de breedte en hoogte van een HTMLLoader-object wijzigt, veranderen de waarden scaleX en scaleY niet, wat wel het geval is bij de meeste andere typen van weergaveobjecten.*

Transparantie van HTMLLoader-inhoud

Adobe AIR 1.0 of hoger

De eigenschap `paintsDefaultBackground` van een HTMLLoader-object, die standaard op `true` is ingesteld, bepaalt of het HTMLLoader-object een ondoorzichtige achtergrond heeft. Als `paintsDefaultBackground` is ingesteld op `false`, is de achtergrond doorzichtig. De weergaveobjectcontainer of andere weergaveobjecten onder het HTMLLoader-object zijn zichtbaar achter de voorgrondelementen van de HTML-inhoud.

Als het hoofdelement of een ander element van het HTML-document een achtergrondkleur opgeeft (bijvoorbeeld via `style="background-color:gray"`), is de achtergrond van het desbetreffende deel van de HTML-inhoud ondoorzichtig en wordt het gerenderd met de opgegeven achtergrondkleur. Als u de eigenschap `opaqueBackground` van het HTMLLoader-object instelt en `paintsDefaultBackground` op `false` staat, is de kleur van `opaqueBackground` zichtbaar.

Opmerking: *U kunt een transparante afbeelding met PNG-indeling gebruiken om een achtergrond met alfa-overvloeiing te bieden voor een element in een HTML-document. Het instellen van de doorzichtigheidsstijl van een HTML-element wordt niet ondersteund.*

HTMLLoader-inhoud schalen

Adobe AIR 1.0 of hoger

Het wordt niet aangeraden een HTMLLoader-object met een factor van meer dan 1,0 te schalen. Tekst in HTMLLoader-inhoud wordt naar een specifieke resolutie gerenderd en vertoont pixels als het HTMLLoader-object te veel wordt geschaald. Als u wilt voorkomen dat het HTMLLoader-object en zijn inhoud worden geschaald bij het vergroten/verkleinen van een venster, stelt u de eigenschap `scaleMode` van Stage in op `StageScaleMode.NO_SCALE`.

Belangrijke opmerkingen voor het laden van SWF- of PDF-inhoud in een HTML-pagina

Adobe AIR 1.0 of hoger

In de volgende gevallen verdwijnt SWF- en PDF-inhoud die in een HTMLLoader-object is geladen:

- als u het HTMLLoader-object schaalt met een andere factor dan 1.0;
- als u de alfa-eigenschap van het HTMLLoader-object instelt op een andere waarde dan 1,0;
- als u de HTMLLoader-inhoud roteert.

De inhoud verschijnt weer als u de conflicterende eigenschapinstelling en de actieve filters verwijdert.

De runtime kan bovendien geen PDF-inhoud in transparante vensters weergeven. De runtime geeft in een HTML-pagina ingesloten SWF-inhoud alleen weer als de parameter `wmode` van het object of de insluitmarkering is ingesteld op `opaque` of `transparent`. De standaardwaarde van `wmode` is`window` en daarom wordt SWF-inhoud alleen weergegeven in transparante vensters als u de parameter wmode expliciet instelt.

Opmerking: *vóór AIR 1.5.2 kon in HTML ingesloten SWF niet worden weergegeven, ongeacht de voor wmode gebruikte waarde.*

Zie "SWF-inhoud insluiten in HTML" op pagina 1051en "PDF-inhoud toevoegen in AIR" op pagina 581 voor meer informatie over het laden van deze mediatypen in een HTMLLoader.

Geavanceerde weergave-eigenschappen

Adobe AIR 1.0 of hoger

De klasse HTMLLoader neemt verschillende methoden over die voor speciale effecten kunnen worden gebruikt. Normaliter hebben deze effecten weliswaar beperkingen wanneer ze worden gebruikt voor HTMLLoader-weergave, maar ze kunnen nuttig zijn voor overgangen of andere tijdelijke effecten. Als u bijvoorbeeld een dialoogvenster weergeeft om gebruikersinvoer op te halen, kunt u de weergave van het hoofdvenster vaag maken tot de gebruiker het dialoogvenster heeft gesloten. Evenzo kunt u de weergave uitfaden bij het sluiten van een venster.

Dit zijn de geavanceerde weergave-eigenschappen:

Eigenschap	Beperkingen
`alpha`	Kan de leesbaarheid van HTML-inhoud verminderen.
`filters`	In een HTML-venster worden buiteneffecten bijgesneden door de vensterrand.
`graphics`	Vormen die met grafische opdrachten worden getekend, verschijnen onder HTML-inhoud, inclusief de standaardachtergrond. Als de getekende vormen zichtbaar moeten zijn, moet de eigenschap paintsDefaultBackground op 'false' staan.
`opaqueBackground`	De kleur van de standaardachtergrond wordt niet gewijzigd. Als deze kleurlaag zichtbaar moet zijn, moet de eigenschap paintsDefaultBackground op 'false' staan.

Eigenschap	Beperkingen
`rotation`	De hoeken van het rechthoekige HTMLLoader-gebied kan door de vensterrand worden bijgesneden. SWF- en PDF-inhoud die in de HTML-inhoud wordt geladen, is niet zichtbaar.
`scaleX`, `scaleY`	De gerenderde weergave kan pixels vertonen bij een schaalfactor van meer dan 1. SWF- en PDF-inhoud die in de HTML-inhoud wordt geladen, is niet zichtbaar.
`transform`	Kan de leesbaarheid van HTML-inhoud verminderen. De HTML-weergave kan door de vensterrand worden bijgesneden. SWF- en PDF-inhoud die in de HTML-inhoud wordt geladen, is niet zichtbaar als roteren, schalen of scheeftrekken wordt toegepast tijdens de transformatie.

In het volgende voorbeeld ziet u hoe u de array `filters` instelt om de volledige HTML-weergave vaag te maken:

```
 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;
```

HTML-inhoud schuiven

Adobe AIR 1.0 of hoger

De klasse HTMLLoader bevat de volgende eigenschappen waarmee u het schuiven van HTML-inhoud kunt bepalen:

Eigenschap	Beschrijving
`contentHeight`	De hoogte van de HTML-inhoud in pixels.
contentWidth	De breedte van de HTML-inhoud in pixels.
scrollH	De horizontale schuifpositie van de HTML-inhoud binnen het HTMLLoader-object.
scrollV	De verticale schuifpositie van de HTML-inhoud binnen het HTMLLoader-object.

De volgende code stelt de eigenschap `scrollV` zo in dat de HTML-inhoud naar de onderkant van de pagina wordt geschoven:

```
 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;
}
```

Het HTMLLoader-object bevat geen horizontale en verticale schuifbalken. U kunt deze desgewenst toevoegen met ActionScript of een Flex-component. De Flex HTML-component bevat automatisch schuifbalken voor HTML-inhoud. U kunt ook de methode `HTMLLoader.createRootWindow()` gebruiken om een venster te maken dat een HTMLLoader-object met schuifbalken bevat (zie "Vensters met schuivende HTML-inhoud maken" op pagina 1079).

Historisch overzicht van HTML openen

Adobe AIR 1.0 of hoger

Wanneer nieuwe pagina's in een HTMLLoader-object worden geladen, worden deze geregistreerd in het historisch overzicht van het object. Dit overzicht komt overeen met het object `window.history` op de HTML-pagina. De klasse HTMLLoader bevat de volgende eigenschappen en methoden waarmee u met het historisch overzicht van HTML kunt werken:

Lid van klasse	Beschrijving
`historyLength`	De totale lengte van het historisch overzicht, inclusief vooruit- en achteruit-items.
`historyPosition`	De huidige positie in het historisch overzicht. Items die vóór deze positie in het overzicht staan, vertegenwoordigen navigatie "achteruit", items na deze positie geven navigatie "vooruit" aan.
`getHistoryAt()`	Geeft het URLRequest-object weer dat overeenkomt met het item op de opgegeven positie in het historisch overzicht.
`historyBack()`	Keert indien mogelijk terug in het historisch overzicht.
`historyForward()`	Gaat indien mogelijk verder in het historisch overzicht.
`historyGo()`	Gaat het opgegeven aantal stappen vooruit of achteruit in het historisch overzicht. Gaat vooruit als de waarde positief is, en keert terug als de waarde negatief is. Wanneer nul wordt bereikt, wordt de pagina opnieuw geladen. Als u een positie verder dan het einde opgeeft, wordt het einde van het overzicht weergegeven.

Items in het geschiedenisoverzicht worden opgeslagen als objecten van het type HTMLHistoryItem. De klasse HTMLHistoryItem heeft de volgende eigenschappen:

Eigenschap	Beschrijving
`isPost`	Zet op `true` als de HTML-pagina POST-gegevens bevat.
`originalUrl`	De oorspronkelijke URL van de HTML-pagina, vóór eventueel doorsturen.
`title`	De titel van de HTML-pagina.
`url`	De URL van de HTML-pagina.

Gebruikersagent voor het laden van HTML-inhoud instellen

Adobe AIR 1.0 of hoger

De klasse HTMLLoader heeft een eigenschap `userAgent`, waarmee u de gebruikersagent-tekenreeks kunt instellen die door de HTMLLoader wordt gebruikt. Stel de eigenschap `userAgent` van het HTMLLoader-object in voordat u de methode `load()` oproept. Als u deze eigenschap op de HTMLLoader-instantie instelt, wordt de eigenschap `userAgent` van de URLRequest die wordt verzonden naar de methode `load()`, *niet* gebruikt.

U kunt de standaardtekenreeks voor de gebruikersagent die door alle HTMLLoader-objecten in een toepassingsdomein wordt gebruikt, instellen via de eigenschap `URLRequestDefaults.userAgent`. De statische URLRequestDefaults-eigenschappen gelden als standaardinstellingen voor alle URLRequest-objecten, niet alleen voor URLRequests die worden gebruikt met de methode `load()` van HTMLLoader-objecten. Wanneer u de eigenschap `userAgent` van een HTMLLoader-object instelt, overschrijft u de standaardinstelling `URLRequestDefaults.userAgent`.

Als u geen gebruikersagent instelt voor de eigenschap `userAgent` van het HTMLLoader-object of voor `URLRequestDefaults.userAgent`, wordt de standaardwaarde van de AIR-gebruikersagent gebruikt. Deze standaardwaarde is afhankelijk van het runtime-besturingssysteem (bijvoorbeeld Mac OS of Windows), de runtime-taal en de runtime, zoals in de volgende twee voorbeelden wordt geïllustreerd:

- `"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"`

Tekencodering voor HTML-inhoud instellen

Adobe AIR 1.0 of hoger

Een HTML-pagina kan de te gebruiken tekencodering instellen door de tag `meta` op te nemen, zoals in het volgende voorbeeld:

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

Overschrijf de pagina-instelling om te zorgen dat een specifieke tekencodering wordt gebruikt. Stel hiervoor de eigenschap `textEncodingOverride` van het HTMLLoader-object in:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

Geef de tekencodering voor HTMLLoader-inhoud op die moet worden gebruikt wanneer een HTML-pagina geen instelling opgeeft. Stel hiervoor de eigenschap `textEncodingFallback` van het HTMLLoader-object in:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

De eigenschap `textEncodingOverride` overschrijft de instelling in de HTML-pagina. En de eigenschap `textEncodingOverride` en de instelling in de HTML-pagina overschrijven de eigenschap `textEncodingFallback`.

Stel de eigenschap `textEncodingOverride` of de eigenschap `textEncodingFallback` in voordat de HTML-inhoud wordt geladen.

Gebruikersinterfaces van het type browser definiëren voor HTML-inhoud

Adobe AIR 1.0 of hoger

JavaScript biedt meerdere API's waarmee u bepaalt in welk venster de HTML-inhoud wordt weergegeven. In AIR kunnen deze API's worden overschreven door de implementatie van een aangepaste HTMLHost-klasse.

Informatie over het uitbreiden van de klasse HTMLHost

Adobe AIR 1.0 of hoger

Als uw toepassing bijvoorbeeld meerdere HTMLLoader-objecten in een venster met tabbladen weergeeft, kunt u zorgen dat titelwijzigingen door de geladen HTML-pagina's worden doorgevoerd in het label van het tabblad, en niet in de titel van het hoofdvenster. Zo kan uw code bijvoorbeeld op een oproep `window.moveTo()` reageren door het HTMLLoader-object in de hoofdweergaveobjectcontainer opnieuw te positioneren door het venster met het HTMLLoader-object te verplaatsen, door niets te doen of door iets helemaal anders te doen.

De AIR-klasse HTMLHost regelt de volgende JavaScript-eigenschappen en -methoden:

- `window.status`
- `window.document.title`
- `window.location`
- `window.blur()`
- `window.close()`
- `window.focus()`
- `window.moveBy()`
- `window.moveTo()`
- `window.open()`
- `window.resizeBy()`
- `window.resizeTo()`

Wanneer u een HTMLLoader-object met behulp van `new HTMLLoader()` maakt, zijn de weergegeven JavaScript-eigenschappen of -methoden niet ingeschakeld. De klasse HTMLHost biedt een standaardimplementatie van deze JavaScript-API's in de vorm van een browser. U kunt ook de klasse HTMLHost uitbreiden om het gedrag aan te passen. Als u een HTMLHost-object wilt maken dat het standaardgedrag ondersteunt, stelt u de parameter `defaultBehaviors` in op 'true' in de HTMLHost-constructor:

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

Als u in AIR een HTML-venster maakt via de methode `createRootWindow()` van de klasse HTMLLoader, wordt automatisch een HTMLHost-instantie toegewezen die de standaardgedragingen ondersteunt. U kunt het gedrag van het hostobject wijzigen door een andere HTMLHost-implementatie toe te wijzen aan de eigenschap `htmlHost` van de HTMLLoader, of u kunt `null` instellen om de voorzieningen volledig uit te schakelen.

Opmerking: *AIR wijst een standaard HTMLHost-object toe aan het beginvenster dat voor een op HTML gebaseerde AIR-toepassing is gemaakt, evenals aan alle vensters die zijn gemaakt via de standaardimplementatie van de JavaScript-methode* `window.open()`.

Voorbeeld: de klasse HTMLHost uitbreiden

Adobe AIR 1.0 of hoger

In het volgende voorbeeld ziet u hoe u door het uitbreiden van de klasse HTMLHost de manier wijzigt waarop een HTMLLoader-object de gebruikersinterface beïnvloedt:

Flex-voorbeeld:

1. Maak een klasse die de klasse HTMLHost uitbreidt (een subklasse).
2. Hef methoden van de nieuwe klasse op om wijzigingen in de instellingen van de gebruikersinterface te verwerken. De volgende klasse, CustomHost, definieert bijvoorbeeld gedragingen voor oproepen van `window.open()` en wijzigingen in `window.document.title`. Oproepen van `window.open()` openen de HTML-pagina in een nieuw venster en wijzigingen in `window.document.title` (zoals de instelling van het element `<title>` van een HTML-pagina) stellen de titel van het desbetreffende venster in.

```
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 Maak in de code die de HTMLLoader bevat (niet de code van de nieuwe subklasse van HTMLHost), een object uit de nieuwe klasse. Wijs het nieuwe object toe aan de eigenschap `htmlHost` van het HTMLLoader-object. De volgende Flex-code gebruikt de klasse CustomHost die u in de vorige stap hebt gedefinieerd:

```
<?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>
```

Voor het testen van deze code neemt u een HTML-bestand met de volgende inhoud op in de toepassingsmap:

```
<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>
```

Flash Professional-voorbeeld:

1. Maak een Flash-bestand voor AIR. Stel de documentklasse in op CustomHostExample en sla het bestand op als CustomHostExample.fla.
2. Maak een ActionScript-bestand met de naam CustomHost.as en zorg dat het een klasse bevat die de klasse HTMLHost uitbreidt (een subklasse). Deze klasse negeert bepaalde methoden van de nieuwe klasse om wijzigingen in de instellingen van de gebruikersinterface te verwerken. De volgende klasse, CustomHost, definieert bijvoorbeeld gedragingen voor oproepen van `window.open()` en wijzigingen in `window.document.title`. Oproepen van de methode `window.open()` openen de HTML-pagina in een nieuw venster en wijzigingen in de eigenschap `window.document.title` (zoals de instelling van het element `<title>` van een HTML-pagina) stellen de titel van het desbetreffende venster in.

```
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 Maak een ander ActionScript-bestand met de naam CustomHostExample.as en zorg dat het de documentklasse voor de toepassing bevat. Deze klasse maakt een HTMLLoader-object en stelt de hosteigenschap van het object in op een instantie van de klasse CustomHost die u in de vorige stap hebt gedefinieerd:

```
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;
        }
    }
}
```

Voor het testen van deze code neemt u een HTML-bestand met de volgende inhoud op in de toepassingsmap:

```
<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>
```

Wijzigingen in de eigenschap window.location verwerken

Adobe AIR 1.0 of hoger

Hef de methode `locationChange()` op om wijzigingen in de URL van de HTML-pagina te verwerken. De methode `locationChange()` wordt opgeroepen wanneer JavaScript in een pagina de waarde van `window.location` wijzigt. In het volgende voorbeeld wordt gewoon de gevraagde URL geladen:

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

Opmerking: *U kunt de eigenschap htmlLoader van het HTMLHost-object gebruiken om naar het huidige HTMLLoader-object te verwijzen.*

JavaScript-oproepen van window.moveBy(), window.moveTo(), window.resizeTo() of window.resizeBy() afhandelen

Adobe AIR 1.0 of hoger

Hef de methode `set windowRect()` op om wijzigingen in de grenzen van de HTML-inhoud af te handelen. De methode `set windowRect()` wordt opgeroepen wanneer JavaScript in een pagina `window.moveBy()`, `window.moveTo()`, `window.resizeTo()` of `window.resizeBy()` oproept. In het volgende voorbeeld worden gewoon de grenzen van het bureaubladvenster bijgewerkt:

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

JavaScript-oproepen van window.open() afhandelen

Adobe AIR 1.0 of hoger

Hef de methode `createWindow()` op om JavaScript-oproepen van `window.open()` af te handelen. Implementaties van de methode `createWindow()` zijn verantwoordelijk voor het maken en retourneren van een nieuw HTMLLoader-object. U geeft de HTMLLoader doorgaans in een nieuw venster weer maar het is niet nodig om een venster te maken.

In het volgende voorbeeld ziet u hoe u de functie `createWindow()` implementeert met behulp van `HTMLLoader.createRootWindow()` om zowel het venster als het HTMLLoader-object te maken. U kunt ook afzonderlijk een NativeWindow-object maken en het HTMLLoader-object toevoegen aan de vensterfase.

```
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;
}
```

Opmerking: *In dit voorbeeld wordt de aangepaste HTMLHost-implementatie toegewezen aan eventuele nieuwe vensters die met `window.open()` zijn gemaakt. U kunt desgewenst ook een andere implementatie gebruiken of de eigenschap `htmlHost` op 'null' instellen voor nieuwe vensters.*

Het object dat als parameter wordt doorgegeven aan de methode `createWindow()` is een HTMLWindowCreateOptions-object. De klasse HTMLWindowCreateOptions bevat eigenschappen die de waarden rapporteren die zijn ingesteld in de parametertekenreeks `features` voor het oproepen van `window.open()`:

HTMLWindowCreateOptions -eigenschap	Overeenkomstige instelling in de functietekenreeks in de JavaScript-oproep van window.open()
`fullscreen`	`fullscreen`
`height`	`height`
`locationBarVisible`	`location`
`menuBarVisible`	`menubar`
`resizeable`	`resizable`
`scrollBarsVisible`	`scrollbars`
`statusBarVisible`	`status`
`toolBarVisible`	`toolbar`
`width`	`width`
`x`	`left` of `screenX`
`y`	`top` of `screenY`

De klasse HTMLLoader implementeert niet alle functies die in de functietekenreeks kunnen worden opgegeven. Uw toepassing moet wanneer nodig schuif-, locatie-, menu-, status- en werkbalken definiëren.

De overige argumenten voor de methode JavaScript `window.open()` worden door het systeem verwerkt. Een implementatie van `createWindow()` laadt normaliter geen inhoud in het HTMLLoader-object en stelt de venstertitel niet in.

JavaScript-oproepen van window.close() afhandelen

Adobe AIR 1.0 of hoger

Hef `windowClose()` op om JavaScript-oproepen van de methode `window.close()` af te handelen. In het volgende voorbeeld wordt het bureaubladvenster gesloten wanneer de methode `window.close()` wordt opgeroepen:

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

JavaScript-oproepen van `window.close()` hoeven het inhoudsvenster niet te sluiten. U kunt bijvoorbeeld het HTMLLoader-object uit de weergavelijst verwijderen zonder het venster (dat nog andere inhoud kan bevatten) te sluiten. Dit wordt geïllustreerd door de volgende code:

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

Wijzigingen in de eigenschap window.status verwerken

Adobe AIR 1.0 of hoger

Hef de methode `updateStatus()` op om JavaScript-wijzigingen in de waarde van `window.status` te verwerken. In het volgende voorbeeld wordt de statuswaarde getraceerd:

```
override public function updateStatus(status:String):void
{
    trace(status);
}
```

De gevraagde status wordt als een tekenreeks verzonden naar de methode `updateStatus()`.

Het HTMLLoader-object zorgt niet voor een statusbalk.

Wijzigingen in de eigenschap window.document.title verwerken

Adobe AIR 1.0 of hoger

Hef de methode `updateTitle()` op om JavaScript-wijzigingen in de waarde van `window.document.title` te verwerken. In het volgende voorbeeld wordt de venstertitel gewijzigd en de tekenreeks "Sample," toegevoegd aan de titel:

```
override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

Als `document.title` is ingesteld op een HTML-pagina, wordt de gevraagde titel als een tekenreeks verzonden naar de methode `updateTitle()`.

Wijzigingen in `document.title` hoeven de titel van het venster met het HTMLLoader-object niet te wijzigen. U kunt bijvoorbeeld een ander interface-element wijzigen, zoals een tekstveld.

JavaScript-oproepen van window.blur() en window.focus() afhandelen

Adobe AIR 1.0 of hoger

Hef de methoden `windowBlur()` en `windowFocus()` op om JavaScript-oproepen van `window.blur()` en `window.focus()` af te handelen, zoals in het volgende voorbeeld wordt geïllustreerd:

```
override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

Opmerking: *AIR biedt geen API voor het uitschakelen van een venster of toepassing.*

Vensters met schuivende HTML-inhoud maken

Adobe AIR 1.0 of hoger

De klasse HTMLLoader bevat de statische methode `HTMLLoader.createRootWindow()`, waarmee u een nieuw venster (vertegenwoordigd door een NativeWindow-object) kunt openen dat een HTMLLoader-object bevat en waarmee u bepaalde gebruikersinterface-instellingen voor het desbetreffende venster kunt definiëren. De methode vereist het instellen van vier parameters, waarmee u de gebruikersinterface kunt definiëren:

Parameter	Beschrijving
`visible`	Een Booleaanse waarde die bepaalt of het venster in eerste instantie zichtbaar is (`true`) of niet (`false`).
`windowInitOptions`	Een NativeWindowInitOptions-object. De klasse NativeWindowInitOptions definieert initialisatieopties voor een NativeWindow-object, zoals de volgende opties: of het venster kan worden geminimaliseerd/gemaximaliseerd of vergroot/verkleind, of het venster de systeeminterface of een aangepaste interface gebruikt, of het venster transparant is (bij vensters die niet de systeeminterface gebruiken), en het type van het venster.
`scrollBarsVisible`	Bepaalt of er schuifbalken zijn (`true`) of niet (`false`).
`bounds`	Een Rectangle-object dat de positie en afmetingen van het nieuwe venster definieert.

In de volgende code wordt bijvoorbeeld de methode `HTMLLoader.createRootWindow()` gebruikt om een venster te maken dat schuifbalken gebruikt en HTMLLoader-inhoud bevat:

```
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();
```

Opmerking: *Vensters die zijn gemaakt door het rechtstreeks in JavaScript oproepen van* `createRootWindow()`*, blijven onafhankelijk van het HTML-venster dat wordt geopend. De JavaScript-venstereigenschappen* `opener` *en* `parent` *bijvoorbeeld staan op* `null`*. Als u* `createRootWindow()` *echter onrechtstreeks oproept door de HTMLHost-methode* `createWindow()` *op te heffen om* `createRootWindow()` *op te roepen, verwijzen* `opener` *en* `parent` *naar het HTML-venster dat wordt geopend.*

Subklassen van de klasse HTMLLoader maken

Adobe AIR 1.0 of hoger

U kunt een subklasse van de klasse HTMLLoader maken om nieuwe gedragingen te creëren. U kunt bijvoorbeeld een subklasse maken die standaardlisteners voor HTMLLoader-gebeurtenissen definieert (zoals de gebeurtenissen die worden verzonden bij het renderen van HTML of wanneer de gebruiker op een koppeling klikt).

In het volgende voorbeeld wordt de klasse HTMLHost uitgebreid om *normaal* gedrag te vertonen wanneer de JavaScript-methode `window.open()` wordt opgeroepen. Vervolgens wordt in het voorbeeld een subklasse van HTMLLoader gedefinieerd die de aangepaste HTMLHost-implementatieklasse gebruikt:

```
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
       }
}
```

De volgende code definieert een subklasse van de klasse HTMLLoader die een MyHTMLHost-object toewijst aan zijn eigenschap `htmlHost`:

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

Zie `Gebruikersinterfaces van het type browser definiëren voor HTML-inhoud` voor meer informatie over de klasse HTMLHost en de methode "Gebruikersinterfaces van het type browser definiëren voor HTML-inhoud" op pagina 1069die in dit voorbeeld wordt gebruikt.

Hoofdstuk 61: Aan HTML gerelateerde gebeurtenissen in AIR afhandelen

Adobe AIR 1.0 of hoger

Met een systeem voor gebeurtenisafhandeling kunnen programmeurs op een gemakkelijke manier reageren op gebruikersinvoer en systeemgebeurtenissen. Het Adobe® AIR®-gebeurtenismodel is niet alleen handig, maar voldoet ook aan de standaarden. Het gebeurtenismodel is gebaseerd op de DOM (Document Object Model) Level 3 Events Specification, een industriestandaard voor de gebeurtenisafhandelingsarchitectuur, en bevat een krachtig en toch intuïtief gebeurtenisafhandelingsmechanisme voor programmeurs.

HTMLLoader-gebeurtenissen

Adobe AIR 1.0 of hoger

Een HTMLLoader-object verzendt de volgende Adobe® ActionScript® 3.0-gebeurtenissen:

Gebeurtenis	Beschrijving
`htmlDOMInitialize`	Wordt verzonden wanneer het HTML-document wordt gemaakt, maar voordat scripts worden geparseerd of DOM-knooppunten worden toegevoegd aan de pagina.
`complete`	Wordt verzonden wanneer het HTML DOM is gemaakt als reactie op een laadbewerking onmiddellijk na de gebeurtenis `onload` in de HTML-pagina.
`htmlBoundsChanged`	Wordt verzonden wanneer een van de eigenschappen `contentWidth` of `contentHeight` (of allebei) is gewijzigd.
`locationChange`	Wordt verzonden wanneer de eigenschap location van de HTMLLoader is gewijzigd.
`locationChanging`	Verzonden voordat de locatie van HTMLLoader verandert door gebruikersnavigatie, een JavaScript-aanroep of een omleiding. De gebeurtenis `locationChanging` wordt niet verzonden wanneer u de methoden `load()`, `loadString()`, `reload()`, `historyGo()`, `historyForward()` of `historyBack()` aanroept. Als u de methode `preventDefault()` van het verzonden gebeurtenisobject aanroept, wordt de navigatie geannuleerd. Als een koppeling wordt geopend in de systeembrowser, wordt er geen locationChanging-gebeurtenis verzonden, aangezien de locatie van HTMLLoader niet verandert.
`scroll`	Wordt altijd verzonden wanneer de HTML-engine de scrollpositie wijzigt. Scroll-gebeurtenissen kunnen plaatsvinden vanwege navigatie naar ankerkoppelingen (#-koppelingen) op de pagina of vanwege het oproepen van de methode `window.scrollTo()`. De gebeurtenis scroll kan ook worden veroorzaakt door het invoeren van tekst in een tekstinvoergebied of tekstgebied.
`uncaughtScriptException`	Wordt verzonden wanneer in de HTMLLoader een JavaScript-uitzondering optreedt waarvoor geen voorziening is getroffen in de JavaScript-code.

U kunt ook een ActionScript-functie registreren voor een JavaScript-gebeurtenis (zoals `onClick`). Zie "DOM-gebeurtenissen afhandelen met ActionScript" op pagina 1083 voor meer informatie.

DOM-gebeurtenissen afhandelen met ActionScript

Adobe AIR 1.0 of hoger

U kunt ActionScript-functies registreren zodat ze reageren op JavaScript-gebeurtenissen. Kijk bijvoorbeeld naar de volgende HTML-inhoud:

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

U kunt een ActionScript-functie registreren als handler voor een willekeurige gebeurtenis in de pagina. Met de volgende code voegt u bijvoorbeeld de functie `clickHandler()` toe als listener voor de gebeurtenis `onclick` van het element `testLink` in de HTML-pagina:

```
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 );
}
```

Het verzonden gebeurtenisobject is niet van het type flash.events.Event of een van de Event-subklassen. Gebruik de Objectklasse om een type van het functieargument van de gebeurtenishandler te declareren.

U kunt ook de methode `addEventListener()` gebruiken om te registreren voor deze gebeurtenissen. U kunt bijvoorbeeld de methode `completeHandler()` in het vorige voorbeeld vervangen door de volgende code:

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

Wanneer een listener verwijst naar een specifiek DOM-element, is het aan te raden te wachten totdat de bovenliggende HTMLLoader de gebeurtenis `complete` heeft verzonden alvorens de gebeurtenislisteners toe te voegen. HTML-pagina's laden vaak meerdere bestanden en het HTML DOM wordt pas volledig samengesteld als alle bestanden zijn geladen en geparseerd. De HTMLLoader verzendt de gebeurtenis `complete` wanneer alle elementen zijn gemaakt.

Reageren op niet-onderschepte JavaScript-uitzonderingen

Adobe AIR 1.0 of hoger

Bekijk de volgende HTML-code:

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

Deze bevat de JavaScript-functie `throwError()`, die verwijst naar de onbekende variabele `melbaToast`:

```
var x = 400 * melbaToast;
```

Als een JavaScript-bewerking een ongeldige bewerking tegenkomt die niet in de JavaScript-code wordt onderschept via een `try`/`catch`-structuur, verzendt het HTMLLoader-object dat de pagina bevat, de gebeurtenis HTMLUncaughtScriptExceptionEvent. U kunt een handler voor deze gebeurtenis registreren, zoals in de volgende code:

```
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);
    }
}
```

Binnen JavaScript kunt u dezelfde gebeurtenis afhandelen via de eigenschap 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>
```

De gebeurtenishandler `htmlErrorHandler()` annuleert het standaardgedrag van de gebeurtenis (dat wil zeggen, het verzenden van een JavaScript-foutbericht naar de AIR trace-output) en genereert een eigen outputbericht. Deze output bevat de waarde van de `exceptionValue` van het HTMLUncaughtScriptExceptionEvent-object. De output bevat de eigenschappen van ieder object in de `stackTrace`-array:

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

Runtimegebeurtenissen afhandelen met JavaScript

Adobe AIR 1.0 of hoger

De runtimeklassen ondersteunen het toevoegen van gebeurtenishandlers met de methode `addEventListener()`. Om een handlerfunctie voor een gebeurtenis toe te voegen, roept u de methode `addEventListener()` op van het object dat de gebeurtenis verzendt. U geeft hierbij het gebeurtenistype en de afhandelingsfunctie op. Als u bijvoorbeeld wilt luisteren naar de gebeurtenis `closing` die wordt verzonden wanneer een gebruiker op het sluitvak van een venster op de titelbalk klikt, gebruikt u de volgende opdracht:

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

Gebeurtenishandlerfuncties maken

Adobe AIR 1.0 of hoger

Met de volgende code maakt u een eenvoudig HTML-bestand dat informatie weergeeft over de positie van het hoofdvenster. Een handlerfunctie met de naam `moveHandler()` luistert naar een verplaatsingsgebeurtenis (gedefinieerd door de klasse NativeWindowBoundsEvent) van het hoofdvenster.

```
<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>
```

Wanneer een gebruiker het venster verplaatst, geven de textarea-elementen de bijgewerkte X- en Y-positie van het venster weer:

Het gebeurtenisobject wordt als argument doorgegeven aan de methode `moveHandler()`. Met de parameter event kan uw handlerfunctie het gebeurtenisobject onderzoeken. In dit voorbeeld gebruikt u de eigenschap `type` van het gebeurtenisobject om te rapporteren dat de gebeurtenis van het type `move` is.

Gebeurtenislisteners verwijderen

Adobe AIR 1.0 of hoger

U kunt de methode `removeEventListener()` gebruiken om een gebeurtenislistener te verwijderen die u niet meer nodig hebt. Het is raadzaam om alle listeners die niet meer worden gebruikt, te verwijderen. Vereiste parameters zijn onder meer de parameters `eventName` en `listener`. Dit zijn dezelfde vereiste parameters als voor de methode `addEventListener()`.

Gebeurtenislisteners verwijderen in HTML-pagina's die navigeren

Adobe AIR 1.0 of hoger

Wanneer HTML-inhoud navigeert, of wanneer HTML-inhoud wordt verwijderd omdat een venster dat deze inhoud bevat wordt gesloten, worden de gebeurtenislisteners die verwijzen naar objecten op de ongeladen pagina, niet automatisch verwijderd. Wanneer een object een gebeurtenis verzendt naar een handler die al is verwijderd, wordt er een foutmelding weergegeven dat de toepassing heeft geprobeerd te verwijzen naar een JavaScript-object in een HTML-pagina die niet meer is geladen.

Om deze fout te voorkomen, verwijdert u JavaScript-gebeurtenislisteners in een HTML-pagina voordat deze verdwijnt. In het geval van paginanavigatie (binnen een HTMLLoader-object) verwijdert u de gebeurtenislistener tijdens de gebeurtenis `unload` van het `window`-object.

De volgende JavaScript-code verwijdert bijvoorbeeld een gebeurtenislistener voor de gebeurtenis `uncaughtScriptException`:

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Om te voorkomen dat de fout optreedt wanneer vensters met HTML-inhoud worden gesloten, roept u een opschoningsfunctie op als reactie op de gebeurtenis `closing` van het NativeWindow-object (`window.nativeWindow`). De volgende JavaScript-code verwijdert bijvoorbeeld een gebeurtenislistener voor de gebeurtenis `uncaughtScriptException`:

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

U kunt deze fout ook voorkomen door een gebeurtenishandler te verwijderen wanneer deze wordt uitgevoerd (als de gebeurtenis slechts eenmaal moet worden uitgevoerd). De volgende JavaScript-code maakt bijvoorbeeld een HTML-venster door de methode `createRootWindow()` van de klasse HTMLLoader op te roepen; hier wordt een gebeurtenislistener voor de gebeurtenis `complete` toegevoegd. Wanneer de gebeurtenishandler `complete` wordt opgeroepen, verwijdert deze zijn eigen gebeurtenislistener met behulp van de functie `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"));
```

Door overbodige gebeurtenislisteners te verwijderen, kan de opschoonfunctie van het systeem ook het geheugen vrijmaken dat is gekoppeld aan die listeners.

Controleren op bestaande gebeurtenislisteners

Adobe AIR 1.0 of hoger

Met de methode `hasEventListener()` kunt u controleren op de aanwezigheid van gebeurtenislisteners op een object.

Hoofdstuk 62: HTML-inhoud weergeven in mobiele toepassingen

Adobe AIR 2.5 of hoger

De klasse StageWebView geeft HTLM-inhoud weer met behulp van het systeembrowserbesturingselement op mobiele apparaten en met gebruik van het standaard Adobe® AIR® HTMLLoader-besturingselement op pc's. Controleer de eigenschap `StageWebView.isSupported` om te bepalen of de klasse wordt ondersteund op het huidige apparaat. Ondersteuning wordt niet gegarandeerd voor alle apparaten in het mobiele profiel.

In alle profielen ondersteunt de klasse StageWebView slechts beperkte interactie tussen de HTML-inhoud en de rest van de toepassing. U kunt navigatie besturen, maar cross-scripting of rechtstreekse gegevensuitwisseling is niet toegestaan. U kunt inhoud laden van een lokale of externe URL of doorgeven in een HTML-reeks.

StageWebView-objecten

Een StageWebView-object is geen weergaveobject en kan niet worden toegevoegd aan de weergavelijst. In plaats daarvan werkt het als een viewport die rechtstreeks aan het werkgebied is gekoppeld. StageWebView-inhoud wordt vóór de inhoud van de weergavelijst getekend. De tekenvolgorde van de verschillende StageWebView-objecten kan niet worden ingesteld.

Als u een StageWebView-object wilt weergeven, wijst u het werkgebied waarop het object moet worden weergegeven toe aan de `stage`-eigenschap van het StageWebView-object. Stel de grootte van de weergave in met de `viewPort`-eigenschap.

Stel de x- en y-coördinaten van de `viewPort`-eigenschap in op tussen -8192 en 8191. De maximale waarde van de breedte en hoogte van het werkgebied is 8191. Als de grootte de maximale waarden overschrijdt, wordt er een uitzondering gegenereerd.

In het volgende voorbeeld wordt een StageWebView-object gemaakt, worden de `stage`- en `viewPort`-eigenschappen ingesteld en wordt een HTML-reeks weergegeven:

```
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 );
```

Als u een StageWebView-object wilt verbergen, stelt u de `stage`-eigenschap in op `null`. Als u het object helemaal wilt vernietigen, roept u de methode `dispose()` aan. Het aanroepen van `dispose()` is optioneel, maar het object wordt dan opgeschoond, zodat het geheugen sneller wordt vrijgegeven.

Inhoud

U kunt met twee methoden inhoud laden in een StageWebView-object: `loadURL()` en `loadString()`.

De `loadURL()`-methode laadt een bron bij de opgegeven URL. U kunt een willekeurig URI-schema gebruiken dat wordt ondersteund door het systeemwebbrowserbesturingselement, zoals: data:, file:, http:, https: en javascript:. De schema's app: en app- storage: worden niet ondersteund. AIR valideert de URL-tekenreeks niet.

De `loadString()`-methode laadt een letterlijke tekenreeks met HTML-inhoud. De locatie van een pagina die met deze methode wordt geladen, wordt uitgedrukt als:

- Op bureaublad: about:blank
- Op iOS: *htmlString*
- Op Android: de gegevens-URI-indeling van de gecodeerde *htmlString*

Het URI-schema bepaalt de regels voor het laden van ingesloten inhoud of gegevens.

URI-schema	Lokale bron laden	Externe bron laden	Lokaal XMLHttpRequest	Extern XMLHttpRequest
data:	Nee	Ja	Nee	Nee
file:	Ja	Ja	Ja	Ja
http:, https:	Nee	Ja	Nee	Hetzelfde domein
about: (loadString()-methode)	Nee	Ja	Nee	Nee

Opmerking: *Als de `displayState`-eigenschap van het werkgebied is ingesteld op `FULL_SCREEN` in Desktop, kan niet in een tekstveld worden getypt dat wordt weergegeven in de StageWebView. In iOS en Android kunt echter in een tekstveld in StageWebView typen, zelfs als de `displayState` van het werkgebied `FULL_SCREEN` is.*

Het volgende voorbeeld gebruikt een StageWebView-object om de Adobe-website weer te geven:

```
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" );
        }
    }
}
```

Op Android-apparaten dient u de Android INTERNET-machtiging op te geven. Anders kan de toepassing geen externe bronnen laden.

In Android 3.0+ moet een toepassing de hardwareversnelling inschakelen in het Android manifestAdditions-element van het beschrijvingsbestand van de AIR-toepassing om de inhoud van de insteekmodule weer te geven in een StageWebView-object. Zie Flash Player en andere insteekmodules inschakelen in een StageWebView-object.

JavaScript-URI

U kunt een JavaScript-URI gebruiken om een functie aan te roepen die is gedefinieerd in de HTML-pagina die wordt geladen door een StageWebView-object. De functie die u aanroept met de JavaScript-URI wordt uitgevoerd in de context van de geladen webpagina. In het volgende voorbeeld wordt een JavaScript-functie aangeroepen door een StageWebView-object:

```
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 );
        }
    }
}
```

Meer Help-onderwerpen

Sean Voisen: Making the Most of StageWebView

Navigatiegebeurtenissen

Wanneer een gebruiker op een koppeling in de HTML klikt, verzendt het StageWebView-object een `locationChanging`-gebeurtenis. Als u de navigatie wilt stoppen, kunt u de methode `preventDefault()` van het gebeurtenisobject aanroepen. Anders navigeert het StageWebView-object naar de nieuwe pagina en wordt een `locationChange`-gebeurtenis verzonden. Wanneer de pagina is geladen, verzendt de StageWebView een `complete`-gebeurtenis.

Een `locationChanging`-gebeurtenis wordt verzonden voor elke HTML-omleiding. De `locationChange`- en `complete`-gebeurtenissen worden op het juiste moment verzonden.

Op iOS wordt een `locationChanging`-gebeurtenis vóór een `locationChange`-gebeurtenis verzonden, behalve voor de eerste `loadURL()`- of `loadString()`-methoden. Er wordt ook een `locationChange`-gebeurtenis verzonden voor navigatiewijzigingen via iFrames en Frames.

Het volgende voorbeeld geeft aan hoe u een locatiewijziging kunt voorkomen en in plaats daarvan een nieuwe pagina kunt openen in de systeembrowser.

```
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 ) );
        }
    }
}
```

Historie

Als een gebruiker op de inhoud klikt die in een StageWebView-object wordt weergegeven, slaat het besturingselement de voor- en achterwaartse historiestapels op. Het volgende voorbeeld toont hoe u door de twee historiestapels kunt navigeren. Het voorbeeld maakt gebruik van de functietoetsen Terug en Zoeken.

```
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();
            }
        }
    }
}
```

Focus

Ook al is het geen weergaveobject, de klasse StageWebView bevat leden die u in staat stellen de focusovergangen van en naar het besturingselement te beheren.

Wanneer het StageWebView-object de focus krijgt, verzendt het een `focusIn`-gebeurtenis. Indien nodig, kunt u deze gebeurtenis gebruiken om de focuselementen in uw toepassing te beheren.

Wanneer de StageWebView de focus opgeeft, verzendt het een `focusOut`-gebeurtenis. Een StageWebView-instantie kan de focus opgeven wanneer een gebruiker voorbij het eerste of laatste besturingselement op de webpagina navigeert met een trackball of de richtingspijlen. De `direction`-eigenschap van het gebeurtenisobject laat u weten of de focusstroom de boven- of de onderzijde van de pagina passeert. Gebruik deze informatie om de focus toe te wijzen aan het desbetreffende weergaveobject boven of onder de StageWebView.

Op iOS kan de focus niet via de programmeertaal worden ingesteld. StageWebView verzendt `focusIn`- en `focusOut`-gebeurtenissen waarbij de direction-eigenschap van `FocusEvent` is ingesteld op `none`. Als de gebruiker het gebied binnen de StageWebView aanraakt, wordt de gebeurtenis `focusIn` verzonden. Als de gebruiker het gebied buiten de StageWebView aanraakt, wordt de gebeurtenis `focusOut` verzonden.

Het volgende voorbeeld toont hoe de focus wordt doorgegeven van het StageWebView-object naar Flash-weergaveobjecten:

```
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;
        }

    }
}
```

Bitmap vastleggen

Een StageWebView-object wordt vóór alle weergavelijstinhoud gerenderd. U kunt geen inhoud toevoegen vóór een StageWebView-object. U kunt een keuzelijst bijvoorbeeld niet uitvouwen vóór de StageWebView-inhoud. U kunt dit probleem oplossen door een opname van de StageWebView vast te leggen. Daarna verbergt u de StageWebView en voegt u in plaats daarvan de bitmapopname toe.

Het volgende voorbeeld toont hoe u de opname van een StageWebView-object vastlegt met gebruik van de `drawViewPortToBitmapData`-methode. Het StageWebView-object wordt verborgen door het werkgebied in te stellen op null. Nadat de webpagina volledig is geladen, wordt een functie aangeroepen die de bitmap vastlegt en weergeeft. Tijdens de uitvoering geeft de code twee labels weer: Google en Facebook. Klik op het label om de corresponderende webpagina vast te leggen en in het werkgebied weer te geven als een opname.

```
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);
        }
    }
}
```

Hoofdstuk 63: Workers voor gelijktijdige uitvoering gebruiken

Flash Player 11.4 of hoger, Adobe AIR 13.4 of hoger voor desktopplatforms

Met ActionScript-workers kunt u code gelijktijdig uitvoeren. Met andere woorden: u kunt code op de achtergrond uitvoeren zonder dat de uitvoering van de hoofdcode wordt onderbroken.

De ActionScript API's voor gelijktijdige uitvoering zijn voor desktopplatforms alleen beschikbaar in Flash Player 11.4 en hoger en in AIR 3.4 en hoger. Gelijktijdige uitvoering wordt niet ondersteund in AIR voor mobiele apparaten.

Workers en gelijktijdige uitvoering

Flash Player 11.4 of hoger, Adobe AIR 13.4 of hoger voor desktopplatforms

Wanneer een toepassing geen worker gebruikt, wordt de toepassingscode stapsgewijs uitgevoerd middels een enkel lineair blok. Dit wordt een *thread* genoemd. De thread voert de code uit die door de ontwikkelaar is geschreven. Naast deze code wordt ook code uitgevoerd die onderdeel is van de runtime. Hierbij gaat het met name om de code waarmee het scherm wordt bijgewerkt nadat de eigenschappen van de weergaveobjecten zijn veranderd. Alhoewel de code is geschreven in blokken die bestaan uit methoden en klassen, wordt de code tijdens runtime regel voor regel uitgevoerd, net als achtereenvolgende stappen in een uitgerekt proces. Neem het volgende hypothetische voorbeeld van stappen die door een toepassing worden uitgevoerd:

1. Enter Frame-gebeurtenis: de runtime roept alle `enterFrame`-gebeurtenishandlers op en voert de bijbehorende coderegels één voor één uit.
2. Mouse-gebeurtenis: de gebruiker verplaatst de muis en de runtime roept alle Mouse-gebeurtenishandlers aan zodra de verschillende rollover- en rollout-gebeurtenissen plaatsvinden.
3. Load Complete-gebeurtenis: het verzoek om een XML-bestand vanuit een URL te laden wordt geretourneerd met de geladen bestandsgegevens. De gebeurtenishandler wordt aangeroepen en doorloopt de benodigde stappen, waarbij de XML-inhoud wordt gelezen en een aantal objecten worden gemaakt op basis van de XML-gegevens.
4. Mouse-gebeurtenis: de muis is weer verplaatst en als gevolg roept de runtime de relevante Mouse-gebeurtenishandlers aan.
5. Rendering: er staan geen gebeurtenissen meer in de wachtrij, zodat de runtime het scherm kan bijwerken op basis van de eventuele wijzigingen die zijn toegepast op weergaveobjecten.
6. Enter Frame-gebeurtenis: de cyclus begint van voor af aan.

Zoals wordt beschreven in het voorbeeld, worden de hypothetische stappen 1-5 achter elkaar uitgevoerd binnen een enkel tijdblok dat een frame wordt genoemd. Aangezien de stappen achtereenvolgens in een enkele thread worden uitgevoerd, kan de runtime geen stap onderbreken om een andere stap uit te voeren. Bij een framesnelheid van 30 frames per seconde moet de runtime alle bewerkingen binnen eendertigste van een seconde uitvoeren. In veel gevallen is dat voldoende tijd om de code uit te voeren en kan de runtime wachten tot de vastgestelde tijdsperiode voor het frame verloopt. Maar stel nu dat de XML-gegevens die in stap 3 worden geladen, bestaan uit een hele grote, diepgeneste

XML-structuur. Dan kan het gebeuren dat de stap waarin de XML-gegevens door de code worden opgehaald en de bijbehorende objecten gemaakt, langer duurt dan eendertigste van een seconde. In dat geval worden de latere stappen(reageren op de muis en vernieuwen van het scherm) later dan verwacht uitgevoerd. Dit leidt tot een schokkerige schermweergave, omdat het scherm niet snel genoeg wordt vernieuwd in reactie op de muisverplaatsing door de gebruiker.

Als alle code binnen dezelfde thread wordt uitgevoerd, kan een dergelijke schokkerige weergave slechts op één manier worden voorkomen. De oplossing? Er eenvoudigweg voor zorgen dat langdurige bewerkingen, zoals het ophalen van een grote gegevensset, niet in de code worden opgenomen. Een andere oplossing wordt geboden door ActionScript-workers. Met workers kunt u langdurige code laten uitvoeren door een aparte worker. Elke worker wordt in een afzonderlijke thread uitgevoerd, zodat de worker de langdurige bewerking in een eigen thread op de achtergrond uitvoert. Dit geeft de uitvoeringsthread van de hoofdworker meer ruimte, zodat het scherm in elk frame probleemloos kan worden vernieuwd.

De mogelijkheid om meerdere codebewerkingen tegelijkertijd uit te voeren wordt *gelijktijdige uitvoering* genoemd. Zodra de worker op de achtergrond zijn codebewerking heeft voltooid (of op 'voortgangspunten' tijdens de uitvoering), kunt u de meldingen en gegevens verzenden naar de worker van de hoofdcode. Op deze manier kunt u code opnemen voor complexe en intensieve bewerkingen, en tegelijkertijd voorkomen dat de gebruiker moet werken met een schokkerige schermweergave.

Workers zijn handig omdat het risico op een tragere framesnelheid doordat de hoofdthread voor rendering wordt geblokkeerd door andere code, wordt verkleind. Workers vereisen echter ook extra systeemgeheugen en CPU-gebruik, en dat gaat soms ten koste van de algehele prestaties van de toepassing. Aangezien elke worker een eigen instantie van de runtime-VM gebruikt, kan zelfs een eenvoudige worker leiden tot een aanzienlijke overhead. Wanneer u workers gebruikt, moet u uw code dan ook testen op alle doelplatforms om te voorkomen dat het systeem niet te zwaar wordt belast. Adobe raadt aan dat u niet meer dan een of twee workers op de achtergrond gebruikt in een typisch scenario.

Workers maken en beheren

Flash Player 11.4 of hoger, Adobe AIR 13.4 of hoger voor desktopplatforms

Als u een worker voor gelijktijdige uitvoering wilt gebruiken, bestaat de eerste stap uit het maken van een worker op de achtergrond. Om een worker te maken zijn twee typen object nodig. Het eerste object is een Worker-instantie. Dit is de worker die u maakt. Het tweede is een WorkerDomain-object. Hiermee wordt de worker gemaakt en worden de Worker-objecten beheerd die in een toepassing worden uitgevoerd.

Het WorkerDomain-object wordt automatisch gemaakt tijdens het laden van de runtime. De runtime maakt ook automatisch een worker voor het SWF-hoofdbestand van de toepassing. Deze eerste worker wordt de *primordial worker* genoemd.

Aangezien er slechts één WorkerDomain-object voor een toepassing is, is de WorkerDomain-instantie toegankelijk via de statische eigenschap `WorkerDomain.current`.

De huidige Worker-instantie (de worker waarin de huidige code wordt uitgevoerd) is op elk gewenst moment toegankelijk via de statische eigenschap `Worker.current`.

Een Worker-object maken van een SWF-bestand

Net zoals de SWF-hoofdcode wordt uitgevoerd door de 'primordial worker', wordt de code van een enkel SWF-bestand uitgevoerd door een worker op de achtergrond. Om een worker op de achtergrond te gebruiken, moet u de code voor de worker schrijven en compileren als een SWF-bestand. Om de worker op de achtergrond te kunnen maken, moeten de bytes van het SWF-bestand als een ByteArray-object toegankelijk zijn voor de bovenliggende worker. Om de worker daadwerkelijk te kunnen maken, geeft u het ByteArray-object door aan de methode `createWorker()` van het WorkerDomain-object.

Er zijn drie manieren waarop de SWF-code van de worker op de achtergrond kan worden opgehaald als ByteArray-object:

Het SWF-bestand van de worker insluiten

Gebruik de [Embed]-metatag om het SWF-bestand van de worker als ByteArray-object in te sluiten in het SWF-hoofdbestand:

```
[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
}
```

Het SWF-bestand van de worker wordt gecompileerd en opgenomen in het SWF-hoofdbestand als een ByteArray-subklasse met de naam BgWorker_ByteClass. Als u een instantie van die klasse maakt, krijgt u een ByteArray-object dat is voorgeladen met de bytes van het SWF-bestand van de worker.

Een extern SWF-bestand van de worker laden

Gebruik een URLLoader-object om een extern SWF-bestand te laden. Het SWF-bestand moet uit hetzelfde beveiligingsdomein stammen, bijvoorbeeld een SWF-bestand dat wordt geladen uit hetzelfde internetdomein als het SWF-hoofdbestand, of het SWF-bestand moet zijn opgenomen in een AIR-toepassingpakket.

```
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
}
```

Wanneer het SWF-bestand volledig is geladen door URLLoader, zijn de bytes van het SWF-bestand beschikbaar in de eigenschap `data` van het URLLoader-object (`event.target.data` in dit voorbeeld).

Het SWF-hoofdbestand gebruiken als het SWF-bestand van de worker

U kunt een enkel SWF-bestand zowel gebruiken als SWF-hoofdbestand en als SWF-bestand voor de worker. Via de eigenschap `loaderInfo.bytes` van de hoofdweergaveklasse hebt u toegang tot de bytes van het SWF-bestand.

```
// 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
    }
}
```

Bij deze methode moet u de code van het SWF-bestand met behulp van een `if`-instructie aftakken binnen de constructor van de hoofdklasse of een methode die wordt aangeroepen. Om te bepalen of de code wordt uitgevoerd in de hoofdworker of in een worker op de achtergrond, kunt u de eigenschap `isPrimordial` van het huidige Worker-object controleren, zoals aangegeven in het voorbeeld.

De uitvoering van een worker starten

Wanneer u de worker hebt gemaakt, kunt u de bijbehorende code uitvoeren door de methode `start()` van het Worker-object aan te roepen. De `start()`-bewerking wordt niet meteen uitgevoerd. Om te controleren of de worker al dan niet wordt uitgevoerd, registreert u een listener voor de gebeurtenis `workerState` van het Worker-object. Die gebeurtenis wordt namelijk verzonden wanneer de status in de levenscyclus van het Worker-object verandert, bijvoorbeeld wanneer het object start met het uitvoeren van code. In uw `workerState`-gebeurtenishandler kunt u controleren of de eigenschap `state` van het Worker-object is ingesteld op `WorkerState.RUNNING`. In dat geval wordt de worker uitgevoerd en is ook de constructor van de bijbehorende hoofdklasse uitgevoerd. De volgende codes laten zien hoe u de registratie voor de `workerState`-gebeurtenis instelt en de methode `start()` aanroept:

```
// 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.
    }
}
```

De uitvoering van een worker beheren

De set met workers die worden uitgevoerd is op elk gewenst moment toegankelijk via de methode `listWorkers()` van de WorkerDomain-klasse. Deze methode retourneert de set met workers waarvan de eigenschap `state` is ingesteld op `WorkerState.RUNNING`, inclusief de 'primordial worker'. Als een worker niet is gestart of al volledig is uitgevoerd, is deze worker niet opgenomen in de set.

Als u een worker niet meer nodig hebt, kunt u de methode `terminate()` van het Worker-object aanroepen om de worker te sluiten en daarbij geheugenruimte en andere systeembronnen vrij te maken.

Communicatie tussen workers

Flash Player 11.4 of hoger, Adobe AIR 13.4 of hoger voor desktopplatforms

Alhoewel workers hun code uitvoeren in afzonderlijke uitvoeringsthreads, zou het weinig voordeel opleveren als dit volledig geïsoleerd gebeurt. Communicatie tussen workers betekent dan ook dat gegevens tussen de workers wordt doorgegeven. Er zijn drie werkwijzen om de gegevens van één worker door te geven aan andere workers.

Als u beslist welke van deze technieken voor gegevensdeling geschikt is voor een specifieke gegevensdoorgiftebehoefte, houdt u rekening met de twee belangrijkste verschillen in deze technieken. Eén verschil is of er al dan niet een gebeurtenis is om de ontvanger te melden dat er nieuwe gegevens beschikbaar zijn en of de ontvangende worker al dan niet controleren op updates. Een ander verschil tussen deze technieken voor gegevensdeling heeft betrekking op de manier waarop de gegevens worden doorgegeven. In sommige gevallen is de ontvangende worker een kopie van de gedeelde gegevens, wat betekent dat er meer objecten worden gemaakt die gebruikmaken van het geheugen en CPU-cycli. In andere gevallen krijgen workers toegang tot objecten die verwijzen naar hetzelfde onderliggende systeemgeheugen, wat betekent dat er minder objecten worden gemaakt en er dus minder geheugen wordt gebruikt. De verschillen worden hier opgesomd:

Communicatietechniek	Verstuurt gebeurtenis bij ontvangst van gegevens	Deelt geheugen tussen workers
Gedeelde eigenschappen tussen workers	Nee	Nee, objecten zijn kopieën, geen verwijzingen
MessageChannel	Ja	Nee, objecten zijn kopieën, geen verwijzingen
Deelbare ByteArray	Nee	Ja, geheugen wordt gedeeld

Gegevens doorgeven met een gedeelde eigenschap

De eenvoudigste manier om gegevens tussen workers te delen is via een gedeelde eigenschap. Elke worker beheert een intern woordenboek met gedeelde eigenschapswaarden. De eigenschappen zijn opgeslagen met sleutelnamen in tekenreeksindeling om ze te onderscheiden van andere eigenschappen. Als u een object in een worker wilt opslaan als gedeelde eigenschap, moet u de methode `setSharedProperty()` van het Worker-object aanroepen met behulp van twee argumenten, de sleutelnaam en de waarde die moet worden opgeslagen:

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

Wanneer de gedeelde eigenschap is ingesteld, kan de waarde worden gelezen door de methode `getSharedProperty()` van het Worker-object aan te roepen en de sleutelnaam door te geven:

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

Het maakt niet uit welke worker de eigenschapswaarde instelt of leest. Zo kan code in een worker op de achtergrond bijvoorbeeld zijn eigen methode `setSharedProperty()` aanroepen om een waarde op te slaan. Vervolgens kan de code die wordt uitgevoerd in de bovenliggende worker, de methode `getSharedProperty()` aanroepen om de gegevens te ontvangen.

Praktisch elk type object kan als waarde worden doorgegeven aan de methode `setSharedProperty()`. Wanneer u de methode `getSharedProperty()` aanroept, bestaat het geretourneerde object uit een kopie van het object dat wordt doorgegeven aan `setSharedProperty()`. Het is dus geen referentie naar hetzelfde object (behalve in een aantal speciale gevallen). Specifieke details over de manier waarop gegevens worden gedeeld, staan beschreven in "Gedeelde referenties en gekopieerde waarden" op pagina 1104.

Het gebruik van een gedeelde eigenschap om gegevens door te geven tussen workers heeft één groot voordeel: de eigenschap is al beschikbaar voordat de worker wordt uitgevoerd. Voordat een worker op de achtergrond wordt uitgevoerd, kan de methode `setSharedProperty()` van het Worker-object van de worker namelijk al worden aangeroepen om een gedeelde eigenschap in te stellen. Wanneer de bovenliggende worker de methode `start()` van de worker aanroept, roept de runtime de constructor aan van de hoofdklasse van de onderliggende worker. Alle gedeelde eigenschappen die zijn ingesteld voordat `start()` werd aangeroepen, zijn beschikbaar om te worden gelezen door de code in de onderliggende worker.

Gegevens doorgeven met een MessageChannel-object

Een MessageChannel-object biedt een methode om gegevens in één richting door te geven tussen twee workers. Het gebruik van een MessageChannel-object voor het doorgeven van gegevens biedt één groot voordeel. Wanneer u een bericht (een object) verzendt via een berichtenkanaal, wordt een `channelMessage`-gebeurtenis verzonden door het MessageChannel-object. De code in de ontvangende worker kan naar die gebeurtenis luisteren om erachter te komen wanneer er gegevens beschikbaar zijn. Op deze manier hoeft de ontvangende worker niet telkens te controleren of de gegevens zijn bijgewerkt.

Een berichtenkanaal vormt een koppeling tussen maximaal twee workers, een zender en een ontvanger. Als u een MessageChannel-object wilt maken, roept u de methode `createMessageChannel()` van het Worker-object van de verzendende worker aan. Hierbij geeft u de ontvangende worker door als argument:

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

Beide workers moeten toegang hebben tot het MessageChannel-object. De eenvoudigste manier om dit te bereiken is het MessageChannel-object doorgeven via de methode `setSharedProperty()`:

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

In de ontvangende worker registreert u een listener voor de gebeurtenis `channelMessage` van het MessageChannel-object. Deze gebeurtenis wordt verzonden wanneer de verzendende worker gegevens verstuurt via het berichtenkanaal.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

Als u ook daadwerkelijk gegevens wilt verzenden, roept u in de verzendende worker de methode `send()` van het MessageChannel-object aan:

```
// In the sending worker swf
sendChannel.send("This is a message");
```

In de ontvangende worker wordt de `channelMessage`-gebeurtenishandler aangeroepen door het MessageChannel-object. Vervolgens kan de ontvangende worker de gegevens ophalen door de methode `receive()` van het MessageChannel-object aan te roepen.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

Het object dat door de receive-methode wordt geretourneerd is van hetzelfde gegevenstype als het object dat werd doorgegeven aan de `send()`-methode. Het ontvangen object is een kopie van het object dat werd doorgegeven door de verzendende worker. Het is dus geen referentie naar het object in de verzendende worker, tenzij het gaat om een van de weinige gegevenstypen die worden beschreven in “Gedeelde referenties en gekopieerde waarden” op pagina 1104.

Gegevens delen met behulp van een deelbare ByteArray

Wanneer een object wordt doorgegeven tussen twee workers, krijgt de ontvangende worker een nieuw object welke een kopie is van het oorspronkelijke object. De twee objecten worden opgeslagen op verschillende locatie in het geheugen van het systeem. Als gevolg hiervan verhoogt elke kopie van het object dat wordt ontvangen, het totale geheugengebruik dat door de runtime wordt gebruikt. Bovendien hebben eventuele wijzigingen die u aanbrengt aan een object in een worker, geen invloed op de kopie in de andere worker. Zie "Gedeelde referenties en gekopieerde waarden" op pagina 1104 voor meer informatie over hoe gegevens worden gekopieerd.

Een ByteArray-object maakt standaard gebruik van hetzelfde gedrag. Als u een ByteArray-instantie doorgeeft aan de methode `setSharedProperty()` van een Worker-object of de methode `send()` van een MessageChannel-object, maakt de runtime een nieuwe ByteArray in het geheugen van de computer en krijgt de ontvangende worker een ByteArray-instantie een verwijzing is naar die nieuwe ByteArray. U kunt dit gedrag echter wijzigen voor een ByteArray-object door de eigenschap `shareable` van dit object in te stellen op `true`.

Als een deelbaar ByteArray-object wordt doorgegeven van de ene worker naar de andere, is de ByteArray-instantie in de ontvangende worker een verwijzing naar hetzelfde onderliggende besturingssysteemgeheugen dat door de ByteArray-instantie in de verzendende worker wordt gebruikt. Als de code in één worker de inhoud van de bytearray wijzigt, worden deze wijzigingen onmiddellijk beschikbaar in andere workers die toegang hebben tot diezelfde gedeelde array.

Omdat workers hun code tegelijk uitvoeren, is het mogelijk dat twee workers op hetzelfde moment toegang proberen te krijgen tot dezelfde bytes in een bytearray. Dit kan leiden tot verlies of beschadiging van gegevens. Er zijn verschillende API's die u kunt gebruiken om de toegang tot gedeelde bronnen te beheren en dergelijke problemen te voorkomen.

De ByteArray-klasse beschikt over methoden waarmee u de inhoud van de bytearray in één bewerking kunt valideren en wijzigen:

- atomicCompareAndSwapIntAt()-methode
- atomicCompareAndSwapLength()-methode

Bovendien bevat het flash.concurrent-pakket klassen die toegangsbeheer bieden wanneer er met gedeelde bronnen wordt gewerkt:

- Mutex-klasse
- Condition-klasse

Gedeelde referenties en gekopieerde waarden

Als u `Worker.setSharedProperty()` of `MessageChannel.send()` aanroept, wordt het object dat wordt doorgegeven aan de ontvangende worker, normaal gesproken via serienummering omgezet in een AMF-indeling. Dit heeft een aantal consequenties:

- Het object dat wordt gemaakt in de ontvangende worker wanneer de methode `getSharedProperty()` van die worker wordt aangeroepen, wordt via serienummering terug omgezet op basis van de AMF-bytes. Dit is een kopie van het oorspronkelijke object en geen referentie naar het object. Wijzigingen die worden doorgevoerd in het object van een van beide workers, worden niet doorgevoerd in de kopie van de andere worker.
- Objecten die niet via serienummering kunnen worden omgezet in een AMF-indeling, zoals weergaveobjecten, kunnen niet worden doorgegeven aan een worker via `Worker.setSharedProperty()` of `MessageChannel.send()`.

- Voor een juiste omzetting via serienummering van een aangepaste klasse moet de klassendefinitie worden geregistreerd met de functie `flash.net.registerClassAlias()` of met `[RemoteClass]`-metagegevens. Hetzelfde alias moet worden gebruikt voor de klassenversies in beide workers.

Er zijn vijf speciale gevallen van objecten die daadwerkelijk worden gedeeld en dus niet tussen workers worden gekopieerd:

- Worker-objecten
- MessageChannel-objecten
- deelbare bytearray (een ByteArray-object waarvan de eigenschap `shareabletrue` is)
- Mutex-objecten
- Condition-objecten

Wanneer u een instantie van een van deze objecten doorgeeft met de methode `Worker.setSharedProperty()` of de methode `MessageChannel.send()`, beschikt elke worker over een referentie naar hetzelfde onderliggende object. Wijzigingen die in één worker worden toegepast op een instantie zijn dan direct beschikbaar in alle andere workers. Bovendien geldt dat als u eenzelfde instantie van een van deze objecten meerdere keren doorgeeft aan een worker, er niet telkens een nieuw exemplaar van het object in de ontvangende worker wordt gemaakt door de runtime. In plaats hiervan wordt dezelfde referentie opnieuw gebruikt.

Overige technieken voor gegevensdeling

Naast de technieken voor het doorgeven van gegevens die specifiek zijn voor workers, kunnen workers ook gegevens uitwisselen via een van de bestaande API's die ondersteuning bieden voor gegevensdeling tussen twee SWF-toepassingen, zoals de volgende:

- lokale, gezamenlijke objecten
- gegevens schrijven naar een bestand in één worker en het bestand lezen in een andere worker
- gegevens opslaan naar en lezen van een SQLite-database

Wanneer een bron wordt gedeeld door twee of meer workers, moet u situaties vermijden waarin de bron gelijktijdig wordt benaderd door meerdere workers. Als een bestand op het lokale bestandssysteem gelijktijdig toegankelijk is voor meerdere workers, kan dat leiden tot gegevensverlies of -beschadiging. Een dergelijk proces wordt mogelijk ook niet ondersteund door het besturingssysteem.

Als beveiliging tegen problemen met gelijktijdige toegang, gebruikt u de Mutex- en Condition-klassen in het flash.concurrent-pakket om toegangsbeheer te bieden wanneer er met gedeelde bronnen wordt gewerkt.

In tegenstelling tot andere mechanismen voor gegevensdeling, is de SQLite-database-engine speciaal ontworpen voor gelijktijdige gegevenstoegang. De engine beschikt over een ingebouwde ondersteuningsfunctie voor transacties. Een SQLite-database is daarom zonder risico op gegevensverlies toegankelijk voor meerdere workers. Aangezien de workers verschillende SQLConnection-instanties gebruiken, krijgt elke worker via een afzonderlijke transactie toegang tot de database. Gelijktijdige bewerkingen voor gegevensmanipulatie zijn dan ook niet van invloed op de gegevensintegriteit.

Zie ook
"Werken met lokale SQL-databases in AIR" op pagina 754
flash.concurrent-pakket

Hoofdstuk 64: Beveiliging

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Beveiliging is voor Adobe, gebruikers, eigenaars van websites en ontwikkelaars van inhoud van zeer groot belang. Daarom zijn er voor Adobe® Flash® Player en Adobe® AIR™ eens et veiligheidsregels en mogelijkheden om de veiligheid van de gebruiker, website-eigenaar en inhoud te waarborgen. Dit onderwerp bevat, tenzij anders vermeld, een uiteenzetting van het beveiligingsmodel voor SWF-bestanden die zijn gepubliceerd met ActionScript 3.0 en worden uitgevoerd in Flash Player 9.0.124.0 of hoger en een uiteenzetting van het beveiligingsmodel voor SWF-, HTML- en JavaScript-bestanden die in AIR 1.0 of hoger worden uitgevoerd.

Deze discussie geeft een overzicht van de beveiliging; er wordt niet geprobeerd om een volledig overzicht te geven van alle implementatiedetails, gebruiksscenario's of onderverdelingen voor het gebruik van bepaalde API's. Zie het Flash Player Developer Center-onderwerp over beveiliging op www.adobe.com/go/devnet_security_nl voor een meer gedetailleerde beschrijving van beveiligingsbegrippen van Flash Player.

Overzicht van beveiliging in het Flash-platform

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een groot deel van het beveiligingsmodel dat door de runtimes van Flash Player en AIR wordt gebruikt, is gebaseerd op het domein waaruit geladen SWF-bestanden, HTML, media en andere elementen afkomstig zijn. Uitvoerbare code in een bestand uit een specifiek internetdomein, zoals www.example.com, heeft altijd toegang tot alle gegevens van dat domein. Deze elementen worden in dezelfde beveiligingsgroep geplaatst, die een *beveiligingssandbox* wordt genoemd. (Zie “Beveiligingssandboxen” op pagina 1108 voor meer informatie.)

ActionScript-code in een SWF-bestand kan bijvoorbeeld SWF-bestanden, bitmaps, audio, tekstbestanden en andere elementen uit het eigen domein laden. Ook is cross-scripting tussen twee SWF-bestanden uit hetzelfde domein altijd toegestaan, mits beide bestanden met ActionScript 3.0 zijn geschreven. *Cross-scripting* is het vermogen van code in het ene bestand om toegang te krijgen tot de eigenschappen, methoden en objecten die door de code in het andere bestand worden gedefinieerd.

Cross-scripting wordt niet ondersteund tussen SWF-bestanden die met ActionScript 3.0 of lager zijn geschreven. Deze bestanden kunnen echter communiceren door middel van de klasse LocalConnection. Verder is het standaard verboden om een SWF-bestand cross-scripting te laten voeren in ActionScript 3.0 SWF-bestanden uit andere domeinen en gegevens uit andere domeinen te laden. Dit soort toegang kan echter wel worden toegestaan met een aanroep van de methode `Security.allowDomain()` in het geladen SWF-bestand. Zie “Cross-scripting” op pagina 1129 voor meer informatie.

De volgende basisregels voor beveiliging zijn standaard van toepassing:

- Bronnen in dezelfde beveiligingssandbox hebben altijd toegang tot elkaar;
- Uitvoerbare code in bestanden in een externe sandbox heeft nooit toegang tot lokale bestanden en gegevens.

De runtimes van Flash Player en AIR beschouwen de volgende vermeldingen als individuele domeinen en hebben voor deze domeinen afzonderlijke beveiligingssandboxen ingesteld:

- `http://example.com`
- `http://www.example.com`

- `http://store.example.com`
- `https://www.example.com`
- `http://192.0.34.166`

Zelfs als een benoemd domein, zoals http://example.com, aan een specifiek IP-adres wordt toegewezen, zoals http://192.0.34.166, stellen de runtimes voor beide afzonderlijke beveiligingssandboxen in.

Een ontwikkelaar kan twee basismethoden gebruiken om ervoor te zorgen dat een SWF-bestand toegang heeft tot elementen van andere sandboxen dan die van het SWF-bestand:

- De methode `Security.allowDomain()` (zie "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120)
- Het URL-beleidsbestand (zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116)

In het beveiligingsmodel van de Flash Player- en AIR-runtimes wordt onderscheid gemaakt tussen het laden van inhoud en het extraheren of ophalen van gegevens. *Inhoud* wordt als media gedefinieerd, zoals visuele media die de runtime kan weergeven, audio, video of een SWF- of HTML-bestand dat weergegeven media bevat. *Gegevens* worden gedefinieerd als iets dat alleen door code toegankelijk is. Inhoud en gegevens worden op verschillende manieren geladen.

- Inhoud laden - U kunt inhoud laden aan de hand van klassen als Loader, Sound en Netstream, aan de hand van MXML-tags als u Flex gebruikt en aan de hand van HTML-tags als u een AIR-toepassing gebruikt.
- Gegevens extraheren: u kunt gegevens uit geladen media extraheren met Bitmap-objecten, de methode `BitmapData.draw()` en `BitmapData.drawWithQuality()`, de eigenschap `Sound.id3` of de methode `SoundMixer.computeSpectrum()`. De methode `drawWithQuality` is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.
- Toegang krijgen tot gegevens - U kunt rechtstreeks toegang krijgen tot gegevens door deze uit een extern bestand te laden (zoals een XML-bestand) met klassen als URLStream, URLLoader, FileReference, Socket en XMLSocket. AIR biedt extra klassen voor het laden van gegevens, bijvoorbeeld de klasse FileStream en de klasse XMLHttpRequest.

Het Flash Player-beveiligingsmodel heeft verschillende regels gedefinieerd voor het laden van inhoud en het benaderen van gegevens. Over het algemeen gelden er minder beperkingen voor het laden van inhoud dan voor het benaderen van gegevens.

Over het algemeen kan inhoud (SWF-bestanden, bitmaps, MP3-bestanden en video's) van een willekeurige plaats worden geladen, maar als de inhoud uit een ander domein afkomstig is dan de code of de inhoud die wordt geladen, wordt de inhoud in een afzonderlijke beveiligingssandbox geplaatst.

Er zijn een aantal obstakels bij het laden van inhoud:

- Lokale SWF-bestanden (degene die van een niet-netwerkadres worden geladen, zoals de vaste schijf van een gebruiker) worden standaard in de sandbox Lokaal-met-bestandssysteem geclassificeerd. Deze bestanden kunnen geen inhoud van het netwerk laden. Zie "Lokale sandboxen" op pagina 1109 voor meer informatie.
- RTMP-servers (Real-Time Messaging Protocol) kunnen toegang tot inhoud beperken. Zie "Inhoud leveren met RTMP-servers" op pagina 1129 voor meer informatie.

Wanneer de geladen media een afbeelding, audio of video is, kunnen de gegevens zoals pixel- en geluidsgegevens alleen worden benaderd door een SWF-bestand dat zich buiten de beveiligingssandbox bevindt als het domein van dat SWF-bestand in een bestand met URL-domeinbeleid was opgenomen in het oorspronkelijke domein van de media. Zie "Geladen media benaderen als gegevens" op pagina 1133 voor meer informatie.

Andere vormen van geladen gegevens zijn tekst- of XML-bestanden die met een object URLLoader worden geladen. Wanneer u gegevens uit een andere beveiligingssandbox wilt benaderen, moet ook hier toestemming worden gegeven aan de hand van een URL-domeinbeleid in het oorspronkelijke domein. Zie "URLLoader en URLStream gebruiken" op pagina 1136 voor meer informatie.

Opmerking: *Beleidsbestanden zijn nooit vereist om code die wordt uitgevoerd in de sandbox van een AIR-toepassing, externe inhoud of gegevens te laten laden.*

Beveiligingssandboxen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Clientcomputers kunnen individuele bestanden met code, inhoud en gegevens uit een aantal bronnen verkrijgen, zoals via externe websites, lokale bestandssystemen of geïnstalleerde AIR-toepassingen. De runtimes van Flash Player en AIR wijzen codebestanden en andere resources (zoals gezamenlijke objecten, bitmaps, geluiden, video's en gegevensbestanden) afzonderlijk aan beveiligingssandboxen toe op basis van hun locatie op het moment dat deze werden geladen. In de volgende secties worden de regels beschreven die door de runtimes worden afgedwongen en die bepalen waartoe code of inhoud die binnen een bepaalde sandbox wordt uitgevoerd, toegang kan krijgen.

Zie het Flash Player Developer Center-onderwerp over beveiliging op www.adobe.com/go/devnet_security_nl voor meer informatie over beveiliging in Flash Player.

Externe sandboxen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De runtimes van Flash Player en AIR plaatsen elementen (waaronder SWF-bestanden) van internet in afzonderlijke sandboxen die dezelfde naam hebben als het domein waaruit ze afkomstig zijn. Elementen die bijvoorbeeld zijn geladen van *example.com*, worden in een andere beveiligingssandbox geplaatst dan elementen die zijn geladen van *foo.org*. Deze bestanden hebben standaard toegang tot alle bronnen van de eigen server. Externe SWF-bestanden kunnen toegang krijgen tot aanvullende gegevens op andere domeinen via expliciete bevoegdheden van de website of maker, zoals URL-beleidsbestanden en de methode `Security.allowDomain()`. Zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116 en "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 voor meer informatie.

Externe SWF-bestanden kunnen geen lokale bestanden of bronnen laden.

Zie het Flash Player Developer Center-onderwerp over beveiliging op www.adobe.com/go/devnet_security_nl voor meer informatie over beveiliging in Flash Player.

Lokale sandboxen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Lokaal bestand beschrijft elk bestand waarnaar wordt verwezen door gebruik van het `bestand:` protocol of een UNC-pad (Universal Naming Convention). Lokale SWF-bestanden worden in een van vier lokale sandboxen geplaatst:

- De sandbox Lokaal-met-bestandssysteem - Om veiligheidsredenen plaatsen de runtimes van Flash Player en AIR alle lokale bestanden standaard in deze sandbox. Vanuit deze sandbox kunnen lokale bestanden door uitvoerbare code worden gelezen (bijvoorbeeld met de klasse URLLoader), maar kan deze code op geen enkele manier communiceren met het netwerk. Op die manier weet de gebruiker zeker dat lokale gegeven niet worden gelekt naar het netwerk of op enige andere manier ongewenst wordt gedeeld.
- De sandbox Lokaal-met-netwerk. Bij het compileren van een SWF-bestand kunt u opgeven dat dit netwerktoegang heeft wanneer het als een lokaal bestand wordt uitgevoerd (zie "Het type sandbox van lokale SWF-bestanden instellen" op pagina 1111). Het bestand wordt dan in deze sandbox geplaatst. SWF-bestanden die zijn toegewezen aan de sandbox Lokaal-met-netwerk hebben geen toegang tot het lokale bestandssysteem. Ze hebben wel toegang tot gegevens op het netwerk. Een SWF-bestand kan in dit geval echter geen gegevens lezen die zijn afgeleid van het netwerk, tenzij daartoe bevoegdheden zijn toegekend, via een bestand met interdomeinbeleid of het aanroepen van de methode `Security.allowDomain()`. Om dergelijke bevoegdheden te kunnen toekennen, moet een bestand met interdomeinbeleid toegang verschaffen aan *alle* domeinen via `<allow-access-from domain="*"/ >` of via `Security.allowDomain("*")`. Zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116 en "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 voor meer informatie.
- De sandbox Lokaal-vertrouwd—Lokale SWF-bestanden die zijn geregistreerd als vertrouwd (door gebruikers of geïnstalleerde programma's) worden in de sandbox Lokaal-vertrouwd geplaatst. Systeembeheerders en gebruikers kunnen uit beveiligingsoverwegingen een lokaal SWF-bestand opnieuw toewijzen, dat wil zeggen uit of naar de sandbox Lokaal-vertrouwd verplaatsen (zie "Controlemiddelen voor beheerders" op pagina 1113 en "Controlemiddelen voor gebruikers" op pagina 1115). SWF-bestanden die zijn toegewezen aan de sandbox Lokaal-vertrouwd , kunnen communiceren met andere SWF-bestanden en gegevens laden (zowel externe als lokale).
- De sandbox van de AIR-toepassing—Deze sandbox bevat inhoud die is geïnstalleerd met de actieve AIR-toepassing. Standaard kan code die wordt uitgevoerd in de sandbox van een AIR-toepassing, vanuit elk gewenst domein cross-scripting uitvoeren. Bestanden buiten de sandbox van een AIR-toepassing hebben echter geen toestemming om cross-scripting op de code in de sandbox van de toepassing toe te passen. Standaard kunnen code en inhoud in de sandbox van een AIR-toepassing vanuit elk gewenst domein inhoud en gegevens laden.

Communicatie tussen de sandbox Lokaal-met-netwerk en de sandbox Lokaal-met-bestandssysteem, alsmede tussen de sandbox Lokaal-met-bestandssysteem en de externe sandboxen, is ten strengste verboden. Er kan geen toestemming voor dergelijke communicatie worden verleend door een toepassing die wordt uitgevoerd in Flash Player of door een gebruiker of beheerder.

Voor scriptbewerkingen tussen lokale HTML-bestanden en lokale SWF-bestanden (in beide richtingen), bijvoorbeeld via de klasse ExternalInterface, moeten zowel het HTML-bestand als het SWF-bestand zich in de sandbox Lokaal-vertrouwd bevinden. De reden hiervoor is dat de lokale-beveiligingsmodellen voor browsers afwijken van het lokale-beveiligingsmodel van Flash Player.

SWF-bestanden in de sandbox Lokaal-met-netwerk kunnen geen SWF-bestanden laden die zich in de sandbox Lokaal-met-bestandssysteem bevinden. SWF-bestanden in de sandbox Lokaal-met-bestandssysteem kunnen geen SWF-bestanden laden die zich in de sandbox Lokaal-met-netwerk bevinden.

De sandbox van AIR-toepassingen

Adobe AIR 1.0 of hoger

Aan het beveiligingssandboxmodel van Flash Player, wordt door de runtime van Adobe AIR nog een sandbox toegevoegd, namelijk de sandbox van de *toepassing*. Bestanden die worden geïnstalleerd als deel van een AIR-toepassing, worden in de sandbox van de toepassing geplaatst. Voor andere bestanden die door de toepassing worden geladen, gelden beveiligingsbeperkingen die overeenkomen met de beperkingen die in het reguliere Flash Player-beveiligingsmodel zijn opgegeven.

Bij de installatie van een toepassing worden alle bestanden in het AIR-pakket geïnstalleerd in een toepassingsmap op de computer van de gebruiker. Ontwikkelaars kunnen in de programmacode naar deze map verwijzen via het URL-schema `app:/` (zie “URI-schema's” op pagina 860). Alle bestanden in de toepassingsmapstructuur worden bij het starten van de toepassing aan de toepassingssandbox toegewezen. De inhoud van de toepassingssandbox heeft alle toegangsrechten die beschikbaar zijn voor een AIR-toepassing, zoals interactie met het lokale bestandssysteem.

Vele AIR-toepassingen gebruiken alleen deze lokaal geïnstalleerde bestanden om de toepassing uit te voeren. AIR-toepassingen zijn echter niet beperkt tot de bestanden in de toepassingsmap; ze kunnen een willekeurig type bestand van een willekeurige bron laden. Dit geldt onder andere voor lokale bestanden op de computer van de gebruiker en bestanden op beschikbare externe bronnen, zoals op een lokaal netwerk of internet. Het type van de bestanden heeft geen invloed op de beveiligingsbeperkingen: geladen HTML-bestanden hebben dezelfde beveiligingsrechten als geladen SWF-bestanden van dezelfde bron.

De inhoud van de toepassingsbeveiligingssandbox heeft toegang tot AIR API's die niet kunnen worden gebruikt door de inhoud van andere sandboxen. Voorbeeld: de eigenschap `air.NativeApplication.nativeApplication.applicationDescriptor`, die de inhoud van het descriptorbestand van de toepassing weergeeft, is beperkt tot de inhoud van de toepassingsbeveiligingssandbox. Een ander voorbeeld van een beperkte API is de klasse FileStream, die methoden voor het lezen van en schrijven naar het lokale bestandssysteem bevat.

ActionScript-API's die alleen beschikbaar zijn voor de inhoud in de beveiligingssandbox van een toepassing, zijn in de *Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform* gemarkeerd met het AIR-logo. Als deze API's in andere sandboxen worden gebruikt, genereert de runtime een SecurityError-fout.

Bij HTML-inhoud (in een HTMLLoader-object) zijn alle AIR JavaScript API's (de API's die beschikbaar zijn via de eigenschap `window.runtime`, of via het object `air` bij het gebruik van het bestand AIRAliases.js) beschikbaar voor inhoud van de toepassingsbeveiligingssandbox. De HTML-inhoud in andere sandboxen heeft geen toegang tot de eigenschap `window.runtime`, zodat deze inhoud geen toegang heeft tot de API's van AIR of Flash.

Voor inhoud die binnen de sandbox van een AIR-toepassing wordt uitgevoerd, gelden de volgende aanvullende beperkingen:

- Voor de HTML-inhoud van de toepassingsbeveiligingssandbox zijn er beperkingen betreffende het gebruik van API's die tekenreeksen dynamisch in programmacode kunnen omzetten nadat de code is geladen. Hierdoor wordt voorkomen dat de toepassing onbedoeld code van niet-toepassingsbronnen (zoals mogelijk onveilige netwerkdomeinen) opneemt (en uitvoert). Een voorbeeld hiervan is het gebruik van de functie `eval()`. Zie “Codebeperkingen voor de inhoud van verschillende sandboxen” op pagina 1152voor meer informatie.
- Om mogelijke phishingaanvallen te blokkeren worden `img` -tags in de HTML-inhoud van ActionScript TextField-objecten genegeerd in de SWF-inhoud van de toepassingsbeveiligingssandbox.
- De inhoud van de toepassingssandbox kan geen gebruik maken van het protocol `asfunction` in de HTML-inhoud van ActionScript 2.0-tekstvelden.

- De SWF-inhoud van de toepassingssandbox kan geen gebruik maken van de cross-domain cache, een nieuwe functie vanaf Flash Player 9 Update 3. Met deze functie kan Flash Player de inhoud van componenten van het Adobe-platform opslaan in een niet-vluchtige cache en vervolgens op aanvraag opnieuw gebruiken in geladen SWF-inhoud (zodat de inhoud niet meerdere keren opnieuw hoeft te worden geladen).

Beperkingen voor JavaScript in AIR

Adobe AIR 1.0 of hoger

In tegenstelling tot de inhoud van de toepassingsbeveiligingssandbox kan JavaScript-inhoud van een niet-toepassingsbeveiligingssandbox *wel* de functie `eval()` aanroepen om op elk gewenst moment dynamisch gegenereerde code uit te voeren. Er gelden in AIR echter wel beperkingen voor JavaScript-inhoud die in een beveiligingssandbox die niet van een toepassing is, wordt uitgevoerd. Deze zijn onder meer:

- JavaScript-code in een niet-toepassingssandbox heeft geen toegang tot het object `window.runtime`, zodat deze code geen AIR API's kan uitvoeren.
- De inhoud van een niet-toepassingsbeveiligingssandbox kan standaard geen XMLHttpRequest-oproepen gebruiken om gegevens te laden van andere domeinen dan het domein waarvan de aanvraag afkomstig is. Toepassingscode kan niet-toepassingsinhoud hiervoor echter toestemming geven door een kenmerk `allowCrossdomainXHR` in te stellen in het containerframe of iframe. Zie "Codebeperkingen voor de inhoud van verschillende sandboxen" op pagina 1152 voor meer informatie.
- Er zijn beperkingen betreffende het oproepen van de JavaScript-methode `window.open()`. Zie "Beperkingen betreffende het oproepen van de JavaScript-methode window.open()" op pagina 1155voor meer informatie.
- De HTML-inhoud van externe (netwerk-) beveiligingssandboxen kan alleen inhoud van het type CSS, `frame`, `iframe` of `img` vanaf externe domeinen (netwerk-URL's) laden.
- De HTML-inhoud van lokaal-met-bestandssysteem, lokaal-met-netwerk of lokaal-vertrouwde sandboxen kan alleen inhoud van het type CSS, `frame`, `iframe` of `img` uit lokale sandboxen (niet van toepassings- of netwerk-URL's) laden.

Zie "Codebeperkingen voor de inhoud van verschillende sandboxen" op pagina 1152voor meer informatie.

Het type sandbox van lokale SWF-bestanden instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een eindgebruiker of de beheerder van een computer kan opgeven dat een lokaal SWF-bestand vertrouwd is, zodat dit bestand gegevens van alle domeinen kan laden, zowel lokaal als op het netwerk. Dit wordt opgegeven in de map Global Flash Player Trust en in de map User Flash Player Trust. Zie "Controlemiddelen voor beheerders" op pagina 1113 en "Controlemiddelen voor gebruikers" op pagina 1115.

Zie "Lokale sandboxen" op pagina 1109 voor meer informatie over lokale sandboxen.

Adobe Flash Professional

U kunt een SWF-bestand voor de lokaal-met-bestandssysteem of de lokaal-met-netwerk sandbox configureren door de publicatie-instellingen van het document in het programma voor het schrijven van programmacode in te stellen.

Adobe Flex
U kunt een SWF-bestand configureren voor de sandbox Lokaal-met-bestandssysteem of voor de sandbox Lokaal-met-netwerk door de markering `use-network` in de Adobe Flex-compiler in te stellen. Zie "About the application compiler options" in *Building and Deploying Adobe Flex 3 Applications* voor meer informatie.

De eigenschap Security.sandboxType

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

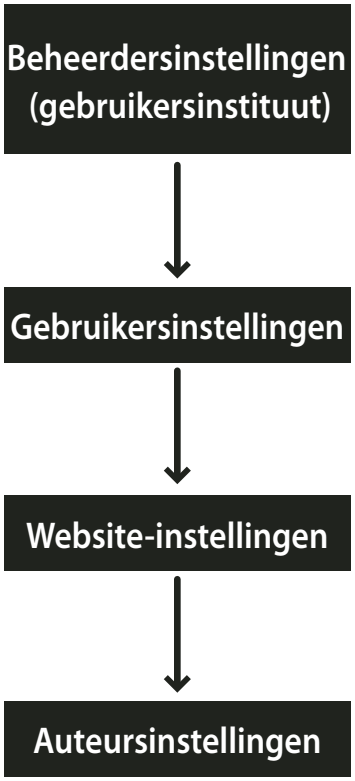
Een maker van een SWF-bestand kan aan de hand van de statische alleen-lezen eigenschap `Security.sandboxType` bepalen aan welk type sandbox de runtime van Flash Player of AIR het SWF-bestand heeft toegewezen. De klasse Security bevat de volgende constanten die mogelijke waarden van de eigenschap `Security.sandboxType` vertegenwoordigen:

- `Security.REMOTE`—Het SWF-bestand is afkomstig van een internet-URL en wordt uitgevoerd volgens op domein gebaseerde sandboxregels.
- `Security.LOCAL_WITH_FILE`—Het SWF-bestand is een lokaal bestand dat niet door de gebruiker als vertrouwd is opgegeven en niet is gepubliceerd met een netwerkbestemming. Het SWF-bestand kan informatie lezen van lokale gegevensbronnen, maar kan niet met internet communiceren.
- `Security.LOCAL_WITH_NETWORK`—Het SWF-bestand is een lokaal bestand dat niet door de gebruiker als vertrouwd is opgegeven, maar wel is gepubliceerd met een netwerkbestemming. Het SWF-bestand kan met internet communiceren maar kan geen informatie lezen van lokale gegevensbronnen.
- `Security.LOCAL_TRUSTED`—Het SWF-bestand is een lokaal bestand dat door de gebruiker als vertrouwd is opgegeven via Settings Manager of een configuratiebestand in de map FlashPlayerTrust. Het SWF-bestand kan informatie lezen van lokale gegevensbronnen en kan met internet communiceren.
- `Security.APPLICATION`— Het SWF-bestand wordt uitgevoerd in een AIR-toepassing en is geïnstalleerd met het pakket (AIR-bestand) voor die toepassing. Standaard kunnen bestanden in de sandbox van de AIR-toepassing cross-scripting uitvoeren vanuit elk gewenst domein. Bestanden buiten de AIR-toepassingssandbox hebben echter geen toestemming om cross-scripting in het AIR-bestand toe te passen. Standaard kunnen bestanden in de sandbox van de AIR-toepassing inhoud en gegevens laden vanuit elk gewenst domein.

Bevoegdheidsbesturingen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het beveiligingsmodel van de Flash Player-client bij uitvoering is rond bronnen ontworpen. Dit zijn objecten zoals SWF-bestanden, lokale gegevens en URL's naar inhoud op internet. *Belanghebbenden* zijn de partijen die eigenaar of gebruiker van deze bronnen zijn. Belanghebbenden kunnen controlemiddelen (beveiligingsinstellingen) toepassen op hun eigen bronnen. Elke bron heeft vier belanghebbenden. Flash Player hanteert een strikte hiërarchie voor de toestemming voor deze controlemiddelen, zoals in de volgende afbeelding wordt getoond:



Hiërarchie van beveiligingscontrolemiddelen

Dit houdt bijvoorbeeld in dat wanneer een beheerder toegang tot een bron beperkt, andere belanghebbenden deze beperking niet kunnen negeren.

Bij AIR-toepassingen zijn deze bevoegdheidscontrolemiddelen alleen van toepassing op inhoud die buiten de sandbox van de AIR-toepassing wordt uitgevoerd.

Controlemiddelen voor beheerders

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een gebruiker met administratieve bevoegdheden van een computer kan beveiligingsbevoegdheden van Flash Player toepassen die effect hebben op alle gebruikers van de computer. In een omgeving anders dan een bedrijf, zoals op een huiscomputer, is er meestal één gebruiker die ook administratieve toegang heeft. Zelfs in een bedrijfsomgeving kunnen afzonderlijke gebruikers administratieve rechten op de computer hebben.

Er zijn twee typen controlemiddelen voor gebruikers met administratieve bevoegdheden:

- Het bestand mms.cfg
- De map Global Flash Player Trust

Het bestand mms.cfg

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Het bestand mms.cfg is een tekstbestand waarmee beheerders toegang tot verschillende functies kunnen toestaan of beperken. Wanneer Flash Player wordt gestart, worden de beveiligingsinstellingen van dit bestand gelezen en worden deze gebruikt om functionaliteit te beperken. Het bestand mms.cfg bevat instellingen waarmee de beheerder functies kan beheren, zoals privacybesturingen, beveiliging van lokale bestanden, socketverbindingen enzovoort.

Een SWF-bestand heeft toegang tot informatie over mogelijkheden die zijn uitgeschakeld door de eigenschappen `Capabilities.avHardwareDisable` en `Capabilities.localFileReadDisable` aan te roepen. De meeste instellingen in het bestand mms.cfg kunnen echter niet vanuit ActionScript worden aangevraagd.

Wanneer u beveiligings- en privacybeleid voor een computer wilt instellen dat niet van een toepassing afhankelijk is, mag het bestand mms.cfg alleen door systeembeheerders worden gewijzigd. Het bestand mms.cfg kan niet door het installatieprogramma van de toepassing worden gebruikt. Hoewel een installatieprogramma dat met administratieve rechten wordt uitgevoerd de inhoud van het bestand mms.cfg kan wijzigen, beschouwt Adobe dergelijk gebruik als een schending van het vertrouwen van de gebruiker en verzoekt de makers van het installatieprogramma dringend om het bestand mms.cfg nooit te wijzigen.

Het bestand mms.cfg wordt op de volgende locatie opgeslagen:

- Windows: *system*\Macromed\Flash\mms.cfg

  (bijvoorbeeld C:\WINDOWS\system32\Macromed\Flash\mms.cfg)

- Mac: *app support*/Macromedia/mms.cfg

  (bijvoorbeeld /Library/Application Support/Macromedia/mms.cfg)

Zie de Flash Player Administration Guide op www.adobe.com/go/flash_player_admin_nl voor meer informatie over het bestand mms.cfg.

De map Global Flash Player Trust

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruikers met administratieve bevoegdheden en installatieprogramma's van toepassingen kunnen specifieke lokale SWF-bestanden als vertrouwd voor alle gebruikers registreren. Deze SWF-bestanden worden aan de sandbox Lokaal-vertrouwd toegekend. Ze kunnen met elk SWF-bestand communiceren en gegevens van willekeurig welke plaats laden, extern of lokaal. Bestanden worden als vertrouwd toegewezen in de map Global Flash Player Trust, op de volgende locatie:

- Windows: *system*\Macromed\Flash\FlashPlayerTrust

  (bijvoorbeeld C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac: *app support*/Macromedia/FlashPlayerTrust

  (bijvoorbeeld /Library/Application Support/Macromedia/FlashPlayerTrust)

De map Flash Player Trust kan een willekeurig aantal tekstbestanden bevatten, waarvan elk bestand het vertrouwde pad weergeeft, met één pad per regel. Elk pad kan een afzonderlijk SWF-bestand, HTML-bestand of map zijn. Opmerkingregels beginnen met het teken `#` . Een configuratiebestand van Flash Player Trust dat de volgende tekst bevat, garandeert bijvoorbeeld vertrouwde status aan alle bestanden in de opgegeven map en aan alle submappen:

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

De paden die in een vertrouwd configuratiebestand worden weergegeven, moeten altijd lokale paden of SMB-netwerkpaden zijn. Alle HTTP-paden in een vertrouwd configuratiebestand worden genegeerd; alleen lokale bestanden kunnen worden vertrouwd.

Geef elk vertrouwensconfiguratiebestand een bestandsnaam die bij de installatietoepassing hoort en gebruik een bestandsextensie .cfg om conflicten te voorkomen.

Wanneer een ontwikkelaar een SWF-bestand dat lokaal wordt uitgevoerd via een installatieprogramma verspreidt, kunt u het installatieprogramma een configuratiebestand laten toevoegen aan de map Global Flash Player Trust, waardoor het bestand dat u verspreidt volledige rechten krijgt. Het installatieprogramma moet worden uitgevoerd door een gebruiker met administratieve rechten. In tegenstelling tot het bestand mms.cfg, wordt de map Global Flash Player Trust opgenomen zodat installatieprogramma's vertrouwensbevoegdheden kunnen toekennen. Zowel gebruikers met administratieve bevoegdheden als installatieprogramma's kunnen lokaal-vertrouwde toepassingen toekennen met de map Global Flash Player Trust.

Er zijn ook mappen Flash Player Trust voor individuele gebruikers (zie "Controlemiddelen voor gebruikers" op pagina 1115).

Controlemiddelen voor gebruikers

Flash Player 9 of hoger

Flash Player bevat op gebruikersniveau drie verschillende mechanismen voor het instellen van bevoegdheden: de Settings-gebruikersinterface en Settings Manager, en de gebruikersmap voor Flash Player-vertrouwen.

De Settings-gebruikersinterface en Settings Manager

Flash Player 9 of hoger

De Settings-gebruikersinterface is een snel en interactief mechanisme voor het configureren van de instellingen voor een specifiek domein. Instellingenbeheer vertegenwoordigt een meer gedetailleerde interface en biedt de mogelijkheid om algemene wijzigingen aan te brengen die effect hebben op de bevoegdheden van veel of van alle domeinen. Wanneer een SWF-bestand een nieuwe bevoegdheid aanvraagt waarbij runtimebeslissingen betreffende beveiliging of privacy zijn vereist, worden bovendien dialoogvensters weergegeven waarin gebruikers bepaalde Flash Player-instellingen kunnen aanpassen.

Instellingenbeheer en Interface voor instellingen bieden beveiligingsopties, zoals instellingen voor camera en microfoon, de opslag van gedeelde objecten, verouderde inhoud, enzovoort. In AIR-toepassingen zijn Settings Manager en de Settings-interface niet beschikbaar.

Opmerking: *De instellingen die in het bestand mms.cfg zijn opgegeven (zie "Controlemiddelen voor beheerders" op pagina 1113) worden niet in Instellingenbeheer weergegeven.*

Zie www.adobe.com/go/settingsmanager_nl voor meer informatie over Instellingenbeheer.

De gebruikersmap voor Flash Player-vertrouwen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gebruikers en installatieprogramma's kunnen specifieke lokale SWF-bestanden als vertrouwd registreren. Deze SWF-bestanden worden aan de sandbox Lokaal-vertrouwd toegekend. Ze kunnen met elk SWF-bestand communiceren en gegevens van willekeurig welke plaats laden, extern of lokaal. Een gebruiker wijst een bestand als vertrouwd toe in de User Flash Player Trust-map, dat zich in dezelfde map bevindt als die van het opslaggebied van de gedeelde objecten (de locatie is afhankelijk van de huidige gebruiker):

- Windows: app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (bijvoorbeeld C:\Documents and Settings\JohnD\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust on Windows XP of C:\Users\JohnD\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust on Windows Vista)

  In Windows is de map Application Data standaard verborgen. Als u verborgen mappen en bestanden wilt weergeven, selecteert u Deze computer om Windows Explorer te openen en vervolgens Extra > Mapopties en opent u het tabblad Weergave. Selecteer op het tabblad Weergave het keuzerondje Verborgen bestanden en mappen weergeven.

- Mac: app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (bijvoorbeeld /Users/JanD/Library/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  Deze instellingen hebben alleen effect op de huidige gebruiker, niet op andere gebruikers die zich op de computer aanmelden. Wanneer een gebruiker zonder administratieve rechten een toepassing installeert in het eigen gedeelte van het systeem, laat de gebruikersmap voor Flash Player-vertrouwen het installatieprogramma de toepassing als vertrouwd registreren voor die gebruiker.

  Wanneer een ontwikkelaar een SWF-bestand dat lokaal wordt uitgevoerd via een installatieprogramma verspreidt, kunt u het installatieprogramma een configuratiebestand laten toevoegen aan de gebruikersmap voor Flash Player-vertrouwen, waardoor het bestand dat u verspreidt volledige rechten krijgt. Zelfs in deze situatie wordt het mapbestand van de User Flash Player Trust als een controlemiddel voor gebruikers beschouwd, omdat een gebruikershandeling (installatie) het initialiseert.

  Er is ook een algemene map voor Flash Player-vertrouwen die door de gebruiker met administratieve bevoegdheden of installatieprogramma's wordt gebruikt om een toepassing voor alle gebruikers van een computer te registreren (zie "Controlemiddelen voor beheerders" op pagina 1113).

Controlemiddelen voor websites (beleidsbestanden)

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt een beleidsbestand op uw webserver maken om gegevens van een webserver beschikbaar te maken voor SWF-bestanden van andere domeinen. Een *beleidsbestand* is een XML-bestand dat op een specifieke locatie op uw server is geplaatst.

Beleidsbestanden hebben effect op de toegang tot een aantal onderdelen, zoals:

- gegevens in bitmaps, geluiden en video's
- het laden van XML- en tekstbestanden;
- SWF-bestanden vanuit andere beveiligingsdomeinen importeren in het beveiligingsdomein van het SWF-bestand dat wordt geladen
- toegang tot socket- en XML-socketverbindingen.

ActionScript-objecten instantiëren twee verschillende soorten serververbindingen: documentgebaseerde serververbindingen en socketverbindingen. ActionScript-objecten zoals Loader, Sound, URLLoader en URLStream instantiëren op document gebaseerde serververbindingen en deze objecten laden een bestand van een URL. ActionScript-objecten Socket en XMLSocket maken socketverbindingen die werken met streaming gegevens, niet met geladen documenten.

Omdat Flash Player twee soorten serververbindingen ondersteunt, zijn er ook twee soorten beleidsbestanden—URL-beleidsbestanden en socketbeleidsbestanden.

- Voor documentgebaseerde verbindingen zijn *URL-beleidsbestanden* vereist. Met deze bestanden kan de server aangeven dat de gegevens en documenten erop beschikbaar zijn voor SWF-bestanden die vanuit sommige of alle domeinen worden bediend.
- Voor socketverbindingen zijn *socketbeleidsbestanden* vereist, waarmee netwerk direct op het lagere TCP-socketniveau wordt ingeschakeld, met behulp van de klassen Socket en XMLSocket.

In Flash Player moeten beleidsbestanden worden verzonden met hetzelfde protocol dat de verbinding wil gebruiken. Wanneer u bijvoorbeeld een beleidsbestand op de HTTP-server plaatst, worden SWF-bestanden van andere domeinen toegestaan hier als een HTTP-server gegevens van te laden. Als u echter geen socketbeleidsbestand op dezelfde server aanbiedt, verbiedt u SWF-bestanden van andere domeinen op socketniveau verbinding met de server te maken. Met andere woorden, de manier waarop een beleidsbestand wordt opgehaald, moet overeenkomen met de manier waarop de verbinding tot stand is gebracht.

Gebruik en syntaxis van beleidsbestanden worden in de rest van deze sectie beknopt beschreven, waar deze van toepassing zijn op SWF-bestanden die voor Flash Player 10 worden gepubliceerd. (De implementatie van beleidsbestanden is iets anders in eerdere versies van Flash Player, aangezien de latere versies van Flash Player de beveiliging is versterkt.) Zie het Flash Player Developer Center-onderwerp over beleidsbestandswijzigingen in Flash Player 9 op www.adobe.com/go/devnet_security_nl voor meer informatie over beveiliging in Flash Player.

Voor code die wordt uitgevoerd in de sandbox van een AIR-toepassing, is geen beleidsbestand vereist voor toegang tot gegevens van een URL of een socket. Voor code van een AIR-toepassing die wordt uitgevoerd in een sandbox die niet van een toepassing is, is wel een beleidsbestand nodig.

Hoofdbeleidsbestanden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Standaard zoekt Flash Player (en AIR-inhoud die zich niet in de AIR-toepassingssandbox bevindt) eerst naar een URL-beleidsbestand met de naam `crossdomain.xml` in de hoofdmap van de server en wordt op poort 843 naar een socketbeleidsbestand gezocht. Als op een van deze locaties een bestand wordt gevonden, wordt dit het *hoofdbeleidsbestand* genoemd. (Bij socketverbindingen zoekt Flash Player ook naar een socketbeleidsbestand op dezelfde poort als de hoofdverbinding. Als er op die poort een beleidsbestand wordt gevonden, wordt dit echter niet als hoofdbeleidsbestand beschouwd.)

Het hoofdbeleidsbestand kan, naast het opgeven van toegangsbevoegdheden, ook een *metabeleid* bevatten. In een metabeleid staat welke locaties beleidsbestanden mogen bevatten. Het standaardmetabeleid voor URL-beleidsbestanden is "master-only", wat betekent dat /crossdomain.xml het enige beleidsbestand is dat op de server is toegestaan. Het standaardmetabeleid voor socketbeleidsbestanden is "all", wat betekent dat elke socket op de host een socketbeleidsbestand mag aanleveren.

Opmerking: *In Flash Player 9 en lager was het standaardmetabeleid voor URL-beleidsbestanden "all", wat betekent dat elke map een beleidsbestand mocht bevatten. Als u toepassingen hebt geïmplementeerd die beleidsbestanden laden uit andere locaties dan het standaardbestand /crossdomain.xml, en die toepassingen kunnen nu in Flash Player 10 worden uitgevoerd, moet u ervoor zorgen (of de serverbeheerder) dat het hoofdbeleidsbestand wordt aangepast, zodat extra beleidsbestanden zijn toegestaan. Zie het Flash Player Developer Center-onderwerp over beleidsbestandswijzigingen in Flash Player 9 op www.adobe.com/go/devnet_security_nl voor meer informatie over het opgeven van een ander metabeleid.*

Een SWF-bestand kan echter op een andere naam of een andere maplocatie controleren door de methode `Security.loadPolicyFile()` aan te roepen. Als het hoofdbeleidsbestand echter niet aangeeft dat de doellocatie beleidsbestanden mag aanleveren, heeft de aanroep naar `loadPolicyFile()` geen effect, zelfs als er op die locatie een beleidsbestand aanwezig is. Roep `loadPolicyFile()` aan voordat u netwerkbewerkingen probeert uit te voeren waarvoor het beleidsbestand vereist is. Flash Player zet netwerkaanvragen automatisch in de wacht achter de pogingen tot benaderen van het bijbehorende beleidsbestand. Het is bijvoorbeeld acceptabel om `Security.loadPolicyFile()` direct vóór het initiëren van een netwerkbewerking op te roepen.

Wanneer Flash Player controleert of er een hoofdbeleidsbestand aanwezig is, wacht het drie seconden op een reactie van de server. Als er geen reactie wordt ontvangen, gaat Flash Player ervan uit dat er geen hoofdbeleidsbestand bestaat. Er is echter geen standaardtime-outwaarde voor aanroepen naar `loadPolicyFile()`. Flash Player gaat ervan uit dat het aangeroepen bestand bestaat en wacht zo lang als nodig is om het te laden. Als u er dus voor wilt zorgen dat er een hoofdbeleidsbestand wordt geladen, moet u `loadPolicyFile()` gebruiken om het expliciet aan te roepen.

De methode heeft de naam `Security.loadPolicyFile()`, maar een beleidsbestand wordt pas geladen wanneer een netwerkaanroep waarvoor een beleidsbestand vereist is, wordt verzonden. Aanroepen naar `loadPolicyFile()` maken Flash Player alleen duidelijk waar naar beleidsbestanden moet worden gezocht wanneer deze nodig zijn.

Het is niet mogelijk om een melding te krijgen wanneer een beleidsbestandaanvraag wordt geïnitieerd of voltooid en er is ook geen reden voor. Flash Player voert beleidscontroles asynchroon uit en wacht automatisch met het initiëren van verbindingen totdat de controles succesvol zijn uitgevoerd.

De volgende secties bevatten informatie die alleen van toepassing is op URL-beleidsbestanden. Zie "Verbinding maken met sockets" op pagina 1136 voor meer informatie over socketbeleidsbestanden.

Bereik van URL-beleidsbestand

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een URL-beleidsbestand is alleen van toepassing op de map waaruit het wordt geladen en op de onderliggende mappen ervan. Een beleidsbestand in de hoofdmap is daarom op de hele server van toepassing, terwijl een beleidsbestand dat vanuit een willekeurige submap wordt geladen alleen op die map en de submappen van toepassing is.

Een beleidsbestand heeft alleen effect op de toegang van een bepaalde server waarop het zich bevindt. Een beleidsbestand dat zich bevindt op https://www.adobe.com:8080/crossdomain.xml is bijvoorbeeld alleen van toepassing op aanroepen voor het laden van gegevens die naar www.adobe.com worden verzonden via het HTTPS-protocol op poort 8080.

Toegangsbevoegdheden opgeven in een URL-beleidsbestand

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een XML-beleidsbestand bevat één `<cross-domain-policy>`-tag, die weer nul of meer `<allow-access-from>`-tags bevat. Elke tag `<allow-access-from>` bevat een attribuut (`domain`) dat een exact IP-adres, een exact domein of een jokertekendomein (elk willekeurig domein) opgeeft. Jokertekendomeinen worden op een van de volgende twee manieren aangegeven:

- Door één sterretje (*), wat overeenkomt met alle domeinen en alle IP-adressen
- Door een sterretje gevolgd door een achtervoegsel, wat overeenkomt met alleen die domeinen die eindigen met het opgegeven achtervoegsel

Achtervoegsels moeten beginnen met een punt. Jokertekendomeinen met achtervoegsels kunnen echter ook staan voor domeinen die bestaan uit uitsluitend het achtervoegsel zonder de voorafgaande punt. xyz.com wordt bijvoorbeeld beschouwd als een onderdeel van *.xyz.com. Jokertekens zijn niet toegestaan in IP-domeinspecificaties.

In het volgende voorbeeld wordt een URL-beleidsbestand weergegeven dat toegang biedt tot SWF-bestanden die van *.example.com , www.friendOfExample.com en 192.0.34.166 afkomstig zijn:

```
<?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>
```

Als u een IP-adres opgeeft, hebben alleen SWF-bestanden toegang die worden geladen vanaf dat IP-adres met IP-syntaxis (bijvoorbeeld http://65.57.83.12/flashmovie.swf). SWF-bestanden die domeinnaamsyntaxis gebruiken, krijgen geen toegang. Flash Player voert geen DNS-omzetting uit.

U kunt toegang toestaan tot documenten die afkomstig zijn van elk willekeurig domein, zoals in het volgende voorbeeld wordt getoond:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

Elke tag `<allow-access-from>` heeft ook het optionele attribuut `secure`, dat standaard op `true` wordt ingesteld. Als het beleidsbestand zich op een HTTPS-server bevindt en u wilt SWF-bestanden op een niet-HTTPS-server toestaan om gegevens van de HTTPS-server te laden, kunt u het attribuut instellen op `false`.

Wanneer u het attribuut `secure` instelt op `false`, brengt u de beveiliging die het HTTPS-protocol u biedt mogelijk in gevaar. Het instellen van dit attribuut op `false` zorgt er vooral voor dat veilige inhoud wordt blootgesteld aan 'snooping-aanvallen' (ongeautoriseerde gegevenstoegang) en 'spoofing-aanvallen' (ongeautoriseerde toegang doordat een onbekende zich als een bekende voordoet). Adobe raadt sterk aan dat u het attribuut `secure` niet op `false` instelt.

Als te laden gegevens zich op een HTTPS-server bevinden, maar het SWF-bestand dat de gegevens laadt zich op een HTTP-server bevindt, raadt Adobe u aan het SWF-bestand naar een HTTPS-server te verplaatsen. Zo houdt u alle kopieën van uw beveiligde gegevens onder de bescherming van HTTPS. Als u echter beslist dat het ladende SWF-bestand op een HTTP-server moet blijven staan, voegt u het attribuut `secure="false"` toe aan de tag `<allow-access-from>`, zoals in de volgende code wordt getoond:

```
<allow-access-from domain="www.example.com" secure="false" />
```

Een ander element dat u kunt gebruiken om toegang te verlenen, is de tag `allow-http-request-headers-from`. Dit element verleent een client met inhoud van een ander toestemmingsdomein toestemming om door de gebruiker gedefinieerde kopteksten naar uw domein te verzenden. Terwijl de tag `<allow-access-from>` andere domeinen toestemming verleent om gegevens uit uw domein te halen, verleent de tag `allow-http-request-headers-from` andere domeinen toestemming om gegevens naar uw domein te verzenden, in de vorm van kopteksten. In het volgende voorbeeld krijgt elk domein toestemming de koptekst SOAPAction naar het huidige domein te verzenden:

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

Als de instructie `allow-http-request-headers-from` in het hoofdbeleidsbestand aanwezig is, is deze van toepassing op alle mappen op de host. Anders is deze alleen van toepassing op de map en submappen van het beleidsbestand dat de instructie bevat.

Beleidsbestanden voorladen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gegevens laden van een server of verbinding maken met een socket is een asynchrone bewerking. Flash Player wacht gewoon totdat het beleidsbestand gedownload is voordat het met de hoofdbewerking begint. Pixelgegevens uit afbeeldingen of samplegegevens uit geluiden extraheren is echter een synchrone bewerking. Het beleidsbestand moet worden geladen voordat u gegevens kunt extraheren. Wanneer u de media laadt, geeft u op dat deze moeten controleren of er een beleidsbestand is:

- Wanneer u de methode `Loader.load()` gebruikt, stelt u de eigenschap `checkPolicyFile` van de parameter `context` in, dat een object LoaderContext is.
- Wanneer u een afbeelding in een tekstveld insluit met de tag `<img>`, stelt u het attribuut `checkPolicyFile` van de tag `<img>` op `"true"` in, zoals hieronder wordt getoond:

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- Wanneer u de methode `Sound.load()` gebruikt, stelt u de eigenschap `checkPolicyFile` van de parameter `context` in, dat een object SoundLoaderContext is.
- Wanneer u de klasse NetStream gebruikt, stelt u de eigenschap `checkPolicyFile` van het object NetStream in.

Wanneer u een van deze parameters instelt, zoekt Flash Player eerst naar beleidsbestanden die reeds voor dat domein zijn gedownload. Vervolgens zoekt het op de standaardlocatie op de server naar het beleidsbestand, waarbij naar instructies `<allow-access-from>` wordt gezocht en naar de aanwezigheid van een metabeleid. Tot slot wordt gekeken naar eventuele lopende aanroepen naar de methode `Security.loadPolicyFile()` om te zien of deze binnen het bereik vallen.

Controlemiddelen voor auteurs (ontwikkelaars)

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

De hoofd-API van ActionScript die wordt gebruikt om beveiligingsrechten toe te staan is de methode `Security.allowDomain()`, die rechten verleent aan SWF-bestanden in de domeinen die u opgeeft. In het volgende voorbeeld verleent een SWF-bestand toegang tot SWF-bestanden die worden aangeboden op het domein www.example.com:

```
Security.allowDomain("www.example.com")
```

Deze methode geeft de volgende machtigingen:

- Scripting tussen SWF-bestanden (zie "Cross-scripting" op pagina 1129)
- Toegang tot het weergaveoverzicht (zie "Het weergaveoverzicht doorlopen" op pagina 1132)
- Gebeurtenisdetectie (zie "Gebeurtenisbeveiliging" op pagina 1132)
- Volledige toegang tot eigenschappen en methoden van het object Stage (zie "werkgebied, beveiliging" op pagina 1131)

Het voornaamste doel van het aanroepen van de methode `Security.allowDomain()` is om SWF-bestanden in een extern domein toe te staan scriptbewerkingen uit te voeren op het SWF-bestand dat de methode `Security.allowDomain()` aanroept. Zie "Cross-scripting" op pagina 1129 voor meer informatie.

Wanneer een IP-adres wordt opgegeven als een parameter voor de methode `Security.allowDomain()`, betekent dit niet dat alle partijen afkomstig van het opgegeven IP-adres toegang hebben. Alleen een partij waarvan het IP-adres specifiek in de URL is opgegeven krijgt toegang, niet een domeinnaam die aan dat IP-adres is toegewezen. Als de domeinnaam www.example.com bijvoorbeeld aan het IP-adres 192.0.34.166 wordt toegewezen, geeft een aanroep van `Security.allowDomain("192.0.34.166")` geen toegang aan www.example.com.

U kunt de joker `"*"` aan de methode `Security.allowDomain()` doorgeven om toegang vanaf alle domeinen toe te staan. Omdat hiermee echter toegang wordt toegestaan aan SWF-bestanden van *alle* domeinen om scriptbewerkingen uit te voeren op het aanroepende SWF-bestand, moet u de joker `"*"` voorzichtig gebruiken.

ActionScript bevat een tweede toestemmings-API met de naam `Security.allowInsecureDomain()`. Deze methode werkt hetzelfde als de methode `Security.allowDomain()` met het verschil dat wanneer de methode vanaf een SWF-bestand wordt aangeroepen dat door een beveiligde HTTPS-verbinding wordt aangeboden, de methode ook toegang tot het aanroepende SWF-bestand toestaat aan andere SWF-bestanden die worden aangeboden vanaf een onbeveiligd protocol, zoals HTTP. Het wordt vanwege de beveiliging echter niet aangeraden om bestanden in een beveiligd protocol (HTTPS) scriptbewerkingen op elkaar uit te laten voeren; als u dit wel toestaat kunt u beveiligde inhoud blootstellen aan snooping- en spoofing-aanvallen. Zo werken dit soort aanvallen: omdat de methode `Security.allowInsecureDomain()` toegang tot uw beveiligde HTTPS-gegevens toestaat aan SWF-bestanden die via HTTP-verbindingen worden aangeboden, kan een aanvaller die zich tussen de verbinding plaatst van de HTTP-server tot de gebruikers het HTTP-SWF-bestand vervangen met een eigen bestand dat vervolgens toegang verkrijgt tot uw HTTPS-gegevens.

Belangrijk: *Code die wordt uitgevoerd in de sandbox van een AIR-toepassing, mag de methoden `allowDomain()` en `allowInsecureDomain()` van de klasse Security niet aanroepen.*

Een andere belangrijke methode met betrekking tot de beveiliging is de methode `Security.loadPolicyFile()`, die ervoor zorgt dat Flash Player naar een bestand met interdomeinbeleid zoekt op een andere locatie dan de standaardlocatie. Zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116 voor meer informatie.

Netwerk-API's beperken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt netwerk-API's op twee manieren beperken. Ter voorkoming van schadelijke activiteiten wordt de toegang tot algemeen gereserveerde poorten geblokkeerd; u kunt deze blokkeringen in uw code niet overschrijven. Als u de toegang van SWF-bestanden tot netwerkfunctionaliteit met betrekking tot andere poorten wilt regelen, kunt u de instelling `allowNetworking` gebruiken.

Geblokkeerde poorten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player en Adobe AIR hebben beperkingen op HTTP-toegang tot bepaalde poorten, net als browsers. HTTP-verzoeken mogen niet op bepaalde standaardpoorten worden gedaan die gewoonlijk worden gebruikt voor niet-HTTP-servers.

API's die toegang verkrijgen tot een netwerk-URL vallen onder deze poortblokkeringsbeperkingen. De enige uitzondering vormen API's die sockets direct aanroepen, zoals `Socket.connect()` en `XMLSocket.connect()`, of aanroepen naar `Security.loadPolicyFile()` waarin een socketbeleidsbestand wordt geladen. Socketverbindingen worden toegestaan of geweigerd via het gebruik van socketbeleidsbestanden op de doelserver.

In het volgende overzicht staan de ActionScript 3.0-API's waarop poortblokkering van toepassing is:

`FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `Loader.loadBytes()`, `navigateToURL()`, `NetConnection.call()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `sendToURL()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`

Poortblokkering is ook van toepassing op het importeren van gedeelde bibliotheken, het gebruik van de tag `<img>` in tekstvelden en het laden van SWF-bestanden in een HTML-pagina met behulp van de tags `<object>` en `<embed>`.

Poortblokkering is ook van toepassing op het gebruik van de tag `<img>` in tekstvelden en het laden van SWF-bestanden op een HTML-pagina die de tags `<object>` en `<embed>` gebruikt.

In het volgende overzicht ziet u welke poorten worden geblokkeerd:

HTTP: 20 (ftp-gegevens), 21 (ftp-controle)

HTTP en 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)

Met behulp van de parameter allowNetworking

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de toegang van een SWF-bestand tot netwerkfunctionaliteit beheren door de parameter `allowNetworking` van de tags `<object>` en `<embed>` in te stellen op de HTML-pagina die de SWF-inhoud bevat.

Mogelijke waarden van `allowNetworking` zijn:

- `"all"` (standaardwaarde) - Alle API's met netwerk hebben toegang tot het SWF-bestand.
- `"internal"` - Het SWF-bestand kan geen API's met browsernavigatie of browserinteractie aanroepen, die verderop in deze sectie worden vermeld, maar kan willekeurig welke andere API met netwerk aanroepen.
- `"none"` - Het SWF-bestand kan geen API's met browsernavigatie of browserinteractie aanroepen, die verderop in deze sectie worden vermeld, en kan ook geen API's gebruiken die van SWF naar SWF communiceren (ook verderop vermeld).

De parameter `allowNetworking` is vooral ontworpen voor gebruik wanneer het SWF-bestand en de HTML-pagina die het bestand bevat uit verschillende domeinen afkomstig zijn. Het gebruik van de waarde `"internal"` of `"none"` is niet aan te raden wanneer het SWF-bestand dat wordt geladen afkomstig is uit hetzelfde domein als de HTML-pagina's die het bestand bevatten, omdat u dan niet kunt waarborgen dat een SWF-bestand altijd met de bedoelde HTML-pagina wordt geladen. Onbetrouwbare partijen kunnen dan een SWF-bestand uit uw domein laden zonder omvattende HTML, waardoor de beperking `allowNetworking` niet werkt zoals u dat hebt bedoeld.

Het aanroepen van een niet-toegestane API genereert een uitzondering SecurityError.

Voeg de parameter `allowNetworking` toe en stel de waarde daarvan in via de tags `<object>` en `<embed>` op de HTML-pagina die een verwijzing bevat naar het SWF-bestand (zie het volgende voorbeeld):

```
<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>
```

Een HTML-pagina kan ook een script gebruiken om tags voor SWF-insluiting te genereren. U moet het script wijzigen, zodat dit de juiste instellingen voor `allowNetworking` invoegt. HTML-pagina's die zijn gegenereerd door Adobe Flash Professional en Adobe Flash Builder gebruiken de `AC_FL_RunContent()`-functie om referenties naar SWF-bestanden in te sluiten. Voeg de `allowNetworking`-parameterinstellingen aan het script toe, zoals hier:

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

De volgende API's zijn niet mogelijk wanneer `allowNetworking` is ingesteld op `"internal"`:

`navigateToURL()`, `fscommand()`, `ExternalInterface.call()`

Naast de bovenstaande API's zijn de volgende API's niet mogelijk wanneer `allowNetworking` is ingesteld op `"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()`

Zelfs wanneer de geselecteerde instelling voor `allowNetworking` toestaat dat een SWF-bestand een API met netwerk gebruikt, kunnen er andere beperkingen gelden, op basis van de beperkingen van de beveiligingssandbox (zie "Beveiligingssandboxen" op pagina 1108).

Wanneer `allowNetworking` is ingesteld op `"none"`, kunt u niet verwijzen naar externe media in een tag `<img>` in de eigenschap `htmlText` van een object TextField (een uitzondering SecurityError wordt dan gegenereerd).

Wanneer `allowNetworking` ingesteld is op `"none"`, wordt een symbool van een geïmporteerde gezamenlijke bibliotheek toegevoegd in de Flash Professional (niet ActionScript) tijdens de runtime geblokkeerd.

Beveiliging in de modus Volledig scherm

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player 9.0.27.0 en hogere versies ondersteunen de modus Volledig scherm, waarbij inhoud die in Flash Player wordt uitgevoerd, het gehele scherm kan vullen. Als u de modus Volledig scherm wilt inschakelen, stelt u de eigenschap `displayState` in op de constante `StageDisplayState.FULL_SCREEN`. Zie "Werken met de modus Volledig scherm" op pagina 177 voor meer informatie.

Voor SWF-bestanden die in een externe sandbox worden uitgevoerd, gelden enkele beveiligingsoverwegingen.

Wanneer u de modus Volledig scherm wilt inschakelen, voegt u de parameter `allowFullScreen` toe aan de tags `<object>` en `<embed>` in de HTML-pagina die een verwijzing naar het SWF-bestand bevat. Hierbij moet allowFullScreen worden ingesteld op `"true"` (de standaardwaarde is `"false"`), zoals in het volgende voorbeeld wordt getoond:

```
<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>
```

Een HTML-pagina kan ook een script gebruiken om tags voor SWF-insluiting te genereren. U moet het script wijzigen, zodat dit de juiste instellingen voor `allowFullScreen invoegt`. HTML-pagina's die zijn gegenereerd door Flash Professional en Flash Builder gebruiken de functie `AC_FL_RunContent()` om verwijzingen naar SWF-bestanden in te sluiten. U moet de instellingen voor de parameter `allowFullScreen` toevoegen, zoals in de volgende code wordt getoond:

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

ActionScript-code voor het inschakelen van de modus Volledig scherm kan alleen worden aangeroepen als reactie op een muisgebeurtenis of toetsenbordgebeurtenis. Als deze code in andere situaties wordt aangeroepen, genereert Flash Player een uitzondering.

Als de inhoud in de modus Volledig scherm wordt weergegeven, wordt een bericht getoond dat aangeeft hoe de gebruiker deze modus kan uitschakelen en naar de normale modus kan terugkeren. Het bericht wordt gedurende enkele seconden weergegeven en fadet dan uit.

Voor inhoud die in een browser wordt uitgevoerd, is het gebruik van het toetsenbord in de modus Volledig scherm beperkt. In Flash Player 9 worden alleen sneltoetsen ondersteund waarmee de toepassing in de normale modus wordt teruggezet, zoals de toets Esc. Gebruikers kunnen geen tekst in tekstvelden invoeren of op het scherm navigeren. In Flash Player 10 en hoger worden bepaalde niet-afdrukbare toetsen (met name de pijltjestoetsen, de spatiebalk en de Tab-toets) ondersteund. Tekstinvoer is echter nog steeds niet toegestaan.

In de zelfstandige speler of in een projectorbestand is de modus Volledig scherm altijd toegestaan. Bovendien wordt het gebruik van het toetsenbord (inclusief tekstinvoer) in deze omgevingen volledig ondersteund.

Wanneer de eigenschap `displayState` van een object Stage wordt aangeroepen, wordt een uitzondering gegenereerd voor elk aanroepend object dat zich niet in dezelfde beveiligingssandbox bevindt als de eigenaar van het werkgebied (het SWF-hoofdbestand). Zie "werkgebied, beveiliging" op pagina 1131 voor meer informatie.

Beheerders kunnen de modus Volledig scherm uitschakelen voor SWF-bestanden die in browsers worden uitgevoerd, door `FullScreenDisable = 1` in te stellen in het bestand mms.cfg. Zie "Controlemiddelen voor beheerders" op pagina 1113 voor meer informatie.

In een browser moet een SWF-bestand zich in een HTML-pagina bevinden om de modus Volledig scherm mogelijk te maken.

Beveiliging in de interactieve schermvullende modus

Flash Player 11.3 of hoger, Adobe AIR 1.0 of hoger

Flash Player 11.3 en hogere versies ondersteunen de interactieve schermvullende modus, waarbij inhoud die in Flash Player wordt uitgevoerd het gehele scherm kan vullen *en bovendien tekstinvoer kan ontvangen.* Als u de interactieve schermvullende modus wilt inschakelen, stelt u de eigenschap `displayState` van het werkgebied in op de constante `StageDisplayState.FULL_SCREEN_INTERACTIVE`. Zie "Werken met de modus Volledig scherm" op pagina 177 voor meer informatie.

Voor SWF-bestanden die in een externe sandbox worden uitgevoerd, gelden enkele beveiligingsoverwegingen.

Wanneer u de modus Volledig scherm wilt inschakelen, voegt u de parameter `allowFullScreenInteractive` toe aan de tags `<object>` en `<embed>` in de HTML-pagina die een verwijzing naar het SWF-bestand bevat. Hierbij moet allowFullScreenInteractive worden ingesteld op `"true"` (de standaardwaarde is `"false"`), zoals in het volgende voorbeeld wordt getoond:

```
<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>
```

Een HTML-pagina kan ook een script gebruiken om tags voor SWF-insluiting te genereren. U moet het script wijzigen, zodat dit de juiste instellingen voor `allowFullScreenInteractive` invoegt. HTML-pagina's die door Flash Professional en Flash Builder zijn gegenereerd, gebruiken de `AC_FL_RunContent()`-functie om referenties in SWF-bestanden in te sluiten en u moet de `allowFullScreenInteractive`-parameterinstellingen toevoegen, zoals hier:

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

ActionScript-code voor het inschakelen van de interactieve schermvullende modus kan alleen worden aangeroepen als reactie op een muisgebeurtenis of toetsenbordgebeurtenis. Als deze code in andere situaties wordt aangeroepen, genereert Flash Player een uitzondering.

Er wordt een bedekkend bericht weergegeven wanneer de interactieve schermvullende modus wordt geactiveerd. In het bericht wordt het domein van de schermvullende pagina weergegeven, naast instructies voor het afsluiten van de schermvullende modus en de knop **Toestaan**. De bedekking verdwijnt pas als de gebruiker op **Toestaan** heeft geklikt om te bevestigen dat de interactieve schermvullende modus is geactiveerd.

Beheerders kunnen de interactieve schermvullende modus uitschakelen voor SWF-bestanden die in browsers worden uitgevoerd door `FullScreenInteractiveDisable = 1` in te stellen in het bestand mms.cfg. Zie "Controlemiddelen voor beheerders" op pagina 1113 voor meer informatie.

In een browser moet een SWF-bestand zich in een HTML-pagina bevinden om de interactieve schermvullende modus mogelijk te maken.

Inhoud laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player- en AIR-inhoud kan vele andere inhoudstypen laden, waaronder:

- SWF-bestanden
- Afbeeldingen
- Geluid
- Video
- HTML-bestanden (alleen AIR)
- JavaScript (alleen AIR)

SWF-bestanden en afbeeldingen met de klasse Loader laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de klasse Loader gebruiken om SWF-bestanden en afbeeldingen te laden (JPG-, GIF- of PNG-bestanden). Alle SWF-bestanden die zich niet in de sandbox Lokaal-met-bestandssysteem bevinden, kunnen SWF-bestanden en afbeeldingen laden vanaf een willekeurig netwerkdomein. Alleen SWF-bestanden in lokale sandboxen kunnen SWF-bestanden en afbeeldingen laden vanaf het lokale bestandssysteem. Bestanden in de sandbox Lokaal-met-netwerk kunnen echter alleen lokale SWF-bestanden laden die zich in de sandbox Lokaal-vertrouwd of Lokaal-met-netwerk bevinden. SWF-bestanden in de sandbox Lokaal-met-netwerk kunnen andere lokale inhoud laden dan SWF-bestanden (zoals afbeeldingen), maar ze krijgen geen toegang tot gegevens in de geladen inhoud.

Wanneer u een SWF-bestand van een niet-vertrouwde bron (zoals een ander domein dan dat van het SWF-hoofdbestand van het object Loader) laadt, kunt u een masker voor het object Loader definiëren om te voorkomen dat de geladen inhoud (die een onderliggend item van het object Loader is) naar delen van het werkgebied buiten het masker tekent, zoals in de volgende code wordt getoond:

```
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);
```

Wanneer u de methode `load()` van het object Loader aanroept, kunt u een parameter `context` opgeven, die een object LoaderContext is. De klasse LoaderContext bevat drie eigenschappen waarmee u kunt definiëren hoe de geladen inhoud wordt gebruikt:

- `checkPolicyFile`: Gebruik deze eigenschap alleen wanneer u een afbeeldingsbestand laadt (geen SWF-bestand). Geef dit op voor een afbeeldingsbestand uit een ander domein dan het domein van het bestand dat het object Loader bevat. Als u deze eigenschap instelt op `true`, controleert de Loader de oorspronkelijke server op een bestand met interdomeinbeleid (zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116). Als de server bevoegdheden toekent aan het domein van Loader, heeft ActionScript-code in SWF-bestanden in het domein Loader toegang tot de gegevens in de geladen afbeelding. Met andere woorden, u kunt de eigenschap `Loader.content` gebruiken om een verwijzing te verkrijgen naar het object Bitmap dat de geladen afbeelding vertegenwoordigt, of u kunt de methode `BitmapData.draw()` of `BitmapData.drawWithQuality()` gebruiken om toegang te krijgen tot de pixels van de geladen afbeelding. De methode `drawWithQuality` is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.
- `securityDomain`: Gebruik deze eigenschap uitsluitend wanneer u een SWF-bestand laadt (geen afbeelding). Geef dit op voor een SWF-bestand uit een ander domein dan het domein van het bestand dat het object Loader bevat. Er worden momenteel slechts twee waarden ondersteund voor de eigenschap `securityDomain`: `null` (standaardwaarde) en `SecurityDomain.currentDomain`. Als u `SecurityDomain.currentDomain` opgeeft, wordt hiermee gevraagd dat het geladen SWF-bestand naar de sandbox van het ladende SWF-bestand wordt *geïmporteerd*. Dit houdt in dat het functioneert alsof het is geladen vanaf de server van het ladende SWF-bestand. Dit is alleen toegestaan als er een URL-beleidsbestand aanwezig is op de server van het geladen SWF-bestand, waardoor toegang mogelijk is voor het domein van het ladende SWF-bestand. Als het vereiste beleidsbestand is gevonden, kunnen lader en geladen inhoud vrij scriptbewerkingen op elkaar uitvoeren als het laden is begonnen, omdat deze zich in dezelfde sandbox bevinden. In plaats van het importeren naar een sandbox kunt u meestal ook een gewone laadbewerking uitvoeren en vervolgens de methode `Security.allowDomain()` laten aanroepen door het geladen SWF-bestand. De laatste methode is mogelijk eenvoudiger, omdat het geladen SWF-bestand zich dan in de eigen natuurlijke sandbox bevindt, en dus toegang kan krijgen tot bronnen op de eigen server.
- `applicationDomain`: Gebruik deze eigenschap uitsluitend wanneer u een SWF-bestand laadt dat in ActionScript 3.0 is geschreven (geen afbeelding of SWF-bestand dat in ActionScript 1.0 of 2.0 is geschreven). Bij het laden van het bestand kunt u opgeven dat het bestand in een bepaald toepassingsdomein wordt geplaatst, en niet, wat standaard het geval is, in een nieuw toepassingsdomein dat een onderliggend domein is van het toepassingsdomein van het ladende SWF-bestand. Toepassingsdomeinen zijn subeenheden van beveiligingsdomeinen. U kunt dus alleen een doeltoepassingsdomein opgeven als het SWF-bestand dat u aan het laden bent, van uw eigen beveiligingsdomein afkomstig is, ofwel omdat dit van uw eigen server komt, ofwel omdat u het met succes in uw beveiligingsdomein hebt geïmporteerd met de eigenschap `securityDomain`. Als u een toepassingsdomein opgeeft, terwijl het geladen SWF-bestand deel uitmaakt van een ander beveiligingsdomein, wordt het domein dat u in `applicationDomain` opgeeft, genegeerd. Zie "Werken met toepassingsdomeinen" op pagina 157 voor meer informatie.

Zie "De context van de geladen gegevens opgeven" op pagina 213 voor meer informatie.

Een belangrijke eigenschap van een object Loader is de eigenschap `contentLoaderInfo`, wat een object LoaderInfo is. Anders dan de meeste andere objecten, wordt een object LoaderInfo tussen het ladende SWF-bestand en de geladen inhoud gedeeld en is het object altijd toegankelijk voor beide partijen. Als de geladen inhoud een SWF-bestand is, heeft het toegang tot het object LoaderInfo via de eigenschap `DisplayObject.loaderInfo`. Objecten LoaderInfo bieden informatie over de voortgang van het laden, de URL's van de lader en geladen inhoud, de vertrouwensrelatie tussen de lader en geladen inhoud en andere informatie. Zie "De voortgang van het laden controleren" op pagina 211 voor meer informatie.

Geluids- en videobestanden laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methoden `Sound.load()`, `NetConnection.connect()` en `NetStream.play()` kunnen geluids- en videobestanden vanaf netwerken worden geladen door alle inhoud, behalve inhoud in de sandbox Lokaal-met-bestandssysteem.

Media van het lokale bestandssysteem kan alleen worden geladen door inhoud in de sandbox Lokaal-met-bestandssysteem en in de sandbox AIR-toepassing. De gegevens in deze geladen bestanden zijn alleen toegankelijk voor inhoud in de sandbox Lokaal-met-bestandssysteem, de sandbox AIR-toepassing en de sandbox Lokaal-vertrouwd.

Er gelden geen andere beperkingen voor het benaderen van gegevens uit geladen media. Zie "Geladen media benaderen als gegevens" op pagina 1133 voor meer informatie.

SWF-bestanden en afbeeldingen in een tekstveld laden met de tag <img >

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt de tag `<img>` gebruiken om SWF-bestanden en bitmaps in een tekstveld te laden, zoals in de volgende code wordt getoond:

```
<img src = 'filename.jpg' id = 'instanceName' >
```

U kunt toegang verkrijgen tot inhoud die zo is geladen met de methode `getImageReference()` van de instantie TextField, zoals in de volgende code wordt getoond:

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

SWF-bestanden en afbeeldingen die zo zijn geladen, worden in de sandbox geplaatst die overeenkomt met de locatie waarvan de bestanden zijn gedownload.

Wanneer u een afbeeldingsbestand in een tekstveld laadt met de tag `<img>`, kunt u mogelijk toegang krijgen tot de gegevens in de afbeelding via een bestand met interdomeinbeleid. U kunt controleren of er een beleidsbestand aanwezig is door een attribuut `checkPolicyFile` aan de tag `<img>` toe te voegen, zoals in de volgende code wordt getoond:

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

Wanneer u een SWF-bestand in een tekstveld laadt met behulp van een tag `<img>`, kunt u toegang tot de gegevens in dat SWF-bestand toestaan via het aanroepen van de methode `Security.allowDomain()`.

Als u een tag `<img>` in een tekstveld gebruikt om een extern bestand te laden (in plaats van een klasse Bitmap die in het SWF-bestand is ingesloten), wordt automatisch een object Loader gemaakt als onderliggend element van het object TextField en wordt het externe bestand in die Loader geladen alsof u een object Loader in ActionScript had gebruikt om het bestand te laden. In dit geval retourneert de methode `getImageReference()` de Loader die automatisch is gemaakt. Er is geen beveiligingscontrole nodig om toegang te krijgen tot dit object Loader, omdat het object zich in dezelfde beveiligingssandbox bevindt als de aanroepende code.

Als u echter via de eigenschap `content` van het object Loader toegang wilt krijgen tot geladen media, zijn beveiligingsregels van toepassing. Als de inhoud een afbeelding is, moet u een bestand met interdomeinbeleid implementeren. Als de inhoud een SWF-bestand is, moet de methode `allowDomain()` worden aangeroepen door de code in het SWF-bestand.

Adobe AIR
In de sandbox van toepassingen worden <img>-tags in tekstvelden genegeerd om phishingaanvallen te voorkomen. Bovendien mag code die in de sandbox van een toepassing wordt uitgevoerd, de methode `allowDomain()` niet aanroepen.

Inhoud leveren met RTMP-servers

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Media Server maakt gebruik van RMTP (Real-Time Media Protocol) bij het leveren van gegevens en geluids- en videobestanden. Deze media kan worden geladen door de methode `connect()` van de klasse NetConnection te gebruiken, waarbij als parameter een RTMP-URL wordt doorgegeven. Flash Media Server kan op basis van het domein van het aanvragende bestand verbindingen beperken en voorkomen dat inhoud wordt gedownload. Bekijk de Flash Media Server-documentatie online op www.adobe.com/go/learn_fms_docs_nl voor meer informatie.

Als u de methoden `BitmapData.draw()`, `BitmapData.drawWithQuality()` en `SoundMixer.computeSpectrum()` wilt gebruiken om runtimeafbeeldingen en geluidsgegevens te extraheren uit RTMP-streams, moet u toegang verlenen tot de server. Met de ActionScript-eigenschappen `Client.videoSampleAccess` en `Client.audioSampleAccess` aan de serverkant kunt u toegang verlenen tot bepaalde mappen op Flash Media Server. Zie de Server-Side ActionScript Language Reference voor meer informatie. (De methode `drawWithQuality` is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.)

Cross-scripting

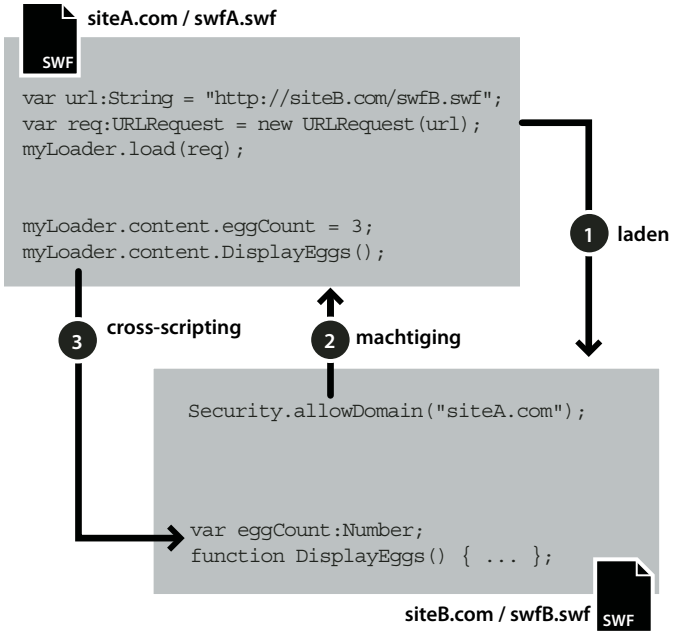
Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als twee SWF-bestanden die met ActionScript 3.0 zijn geschreven of twee HTML-bestanden die in AIR worden uitgevoerd, afkomstig zijn uit hetzelfde domein (bijvoorbeeld de URL voor het ene SWF-bestand is http://www.example.com/swfA.swf en voor het andere http://www.example.com/swfB.swf), kan code die in het ene bestand is gedefinieerd variabelen, objecten, eigenschappen, methoden enzovoort in het andere bestand lezen en wijzigen en vice versa. Dit wordt *cross-scripting* genoemd.

Als de twee bestanden afkomstig zijn uit verschillende domeinen (bijvoorbeeld http://siteA.com/swfA.swf en http://siteB.com/swfB.swf) geven Flash Player en AIR standaard geen toestemming aan swfA.swf om scriptbewerkingen uit te voeren op swfB.swf en ook niet aan swfB.swf om scriptbewerkingen uit te voeren op swfA.swf. Een SWF-bestand geeft toestemming aan SWF-bestanden in andere domeinen door `Security.allowDomain()` aan te roepen. Door `Security.allowDomain("siteA.com")` aan te roepen geeft swfB.swf toestemming aan SWF-bestanden van siteA.com om scriptbewerkingen uit te voeren op swfB.swf.

Cross-scripting wordt niet ondersteund tussen AVM1 SWF-bestanden en AVM2 SWF-bestanden. Een AVM1 SWF-bestand is gemaakt met ActionScript 1.0 of ActionScript 2.0. (AVM1 en AVM2 verwijzen naar de virtuele ActionScript-machine.) U kunt echter de klasse LocalConnection gebruiken om gegevens tussen AVM1 en AVM2 te verzenden.

In elke interdomeinsituatie is het belangrijk om duidelijk te zijn over de rol van de twee betrokken partijen. De partij die cross-scripting uitvoert, wordt de *benaderende partij* genoemd (doorgaans het benaderende SWF-bestand) en de andere partij wordt de *benaderde partij* genoemd (doorgaans het SWF-bestand dat wordt geopend). Als siteA.swf scriptbewerkingen uitvoert op siteB.swf, is siteA.swf de benaderende partij en siteB.swf de benaderde partij, zoals in de volgende afbeelding wordt getoond:



Interdomeinbevoegdheden die met `Security.allowDomain()` worden ingesteld, zijn asymmetrisch. In het vorige voorbeeld kan siteA.swf scriptbewerkingen uitvoeren op siteB.swf, maar niet andersom, omdat siteA.swf de methode `Security.allowDomain()` niet heeft aangeroepen om SWF-bestanden op siteB.com toestemming te geven scriptbewerkingen op siteA.swf uit te voeren. U kunt symmetrische bevoegdheden instellen door beide SWF-bestanden de methode `Security.allowDomain()` te laten aanroepen.

Naast het beveiligen van SWF-bestanden tegen interdomein-scripting door andere SWF-bestanden, beveiligt Flash Player SWF-bestanden tegen interdomein-scripting door HTML-bestanden. HTML-naar-SWF-scriptbewerkingen kunnen optreden met callback-bewerkingen door de methode `ExternalInterface.addCallback()`. Wanneer HTML-naar-SWF-scriptbewerkingen in verschillende domeinen worden uitgevoerd, moet het SWF-bestand dat wordt benaderd de methode `Security.allowDomain()` aanroepen, net als wanneer de benaderende partij een SWF-bestand is, anders mislukt de bewerking. Zie "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 voor meer informatie.

Flash Player biedt ook beveiligingscontroles voor SWF-naar-HTML-scriptbewerkingen. Zie "Uitgaande URL-toegang beheren" op pagina 1140 voor meer informatie.

werkgebied, beveiliging

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Sommige eigenschappen en methoden van het object Stage zijn beschikbaar voor elke sprite of filmclip in het weergaveoverzicht.

Het object Stage heeft echter een eigenaar: het eerste geladen SWF-bestand. De volgende eigenschappen en methoden van het object Stage zijn standaard alleen beschikbaar voor SWF-bestanden die zich in dezelfde beveiligingssandbox bevinden als de eigenaar van het werkgebied:

Eigenschappen	Methoden
`uitlijnen`	`addChild()`
displayState	`addChildAt()`
frameRate	`addEventListener()`
`height`	`dispatchEvent()`
`mouseChildren`	`hasEventListener()`
`numChildren`	`setChildIndex()`
`kwaliteit`	`willTrigger()`
`scaleMode`	
`showDefaultContextMenu`	
`stageFocusRect`	
`stageHeight`	
`stageWidth`	
`tabChildren`	
`textSnapshot`	
`width`	

Als een SWF-bestand dat zich in een andere sandbox bevindt als de eigenaar van het werkgebied, toegang wil verkrijgen tot deze eigenschappen en methoden, moet het SWF-bestand van de eigenaar van het werkgebied de methode `Security.allowDomain()` aanroepen om het domein van de externe sandbox toe te staan. Zie "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 voor meer informatie.

De eigenschap `frameRate` is een speciaal geval: elk SWF-bestand kan de eigenschap `frameRate` lezen. Alleen de bestanden in de beveiligingssandbox van de eigenaar van het werkgebied (of bestanden die bevoegdheden hebben verkregen via het aanroepen van de methode `Security.allowDomain()`) kunnen de eigenschap echter wijzigen.

Er bestaan ook beperkingen voor de methoden `removeChildAt()` en `swapChildrenAt()` van het object Stage, maar deze zijn anders dan de andere beperkingen. Om deze methoden aan te kunnen roepen, moet deze code zich ofwel in hetzelfde domein bevinden als de eigenaar van de betreffende onderliggende object(en) (in plaats van in hetzelfde domein als de eigenaar van het werkgebied), ofwel moet de methode `Security.allowDomain()` door de betreffende objecten worden aangeroepen.

Het weergaveoverzicht doorlopen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Een SWF-bestand heeft beperkte mogelijkheden om toegang te krijgen tot weergaveobjecten die zijn geladen uit andere sandboxen. Een SWF-bestand heeft alleen toegang tot een weergaveobject dat door een ander SWF-bestand in een andere sandbox is gemaakt als het benaderde SWF-bestand de methode `Security.allowDomain()` aanroept om het domein van het SWF-bestand dat het object wil gebruiken toegang te bieden. Zie "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 voor meer informatie.

Toegang tot een object Bitmap dat is geladen door een object Loader is alleen mogelijk als er op de oorspronkelijke server van het afbeeldingsbestand een bestand met interdomeinbeleid is gedefinieerd en in dit bestand toestemming wordt verleend aan het domein van het SWF-bestand dat het object Bitmap wil gebruiken (zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116).

Het object LoaderInfo dat overeenkomt met een geladen bestand (en met het object Loader) bevat de volgende drie eigenschappen, die bepalend zijn voor de relatie tussen het geladen object en het object Loader: `childAllowsParent`, `parentAllowsChild` en `sameDomain`.

Gebeurtenisbeveiliging

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er gelden beveiligingsbeperkingen voor de toegang tot gebeurtenissen die betrekking hebben op het weergaveoverzicht. Deze beperkingen zijn gebaseerd op de sandbox van het weergaveobject dat de gebeurtenis verzendt. Een gebeurtenis in het weergaveoverzicht heeft vastleg- en terugkoppelfasen (zie "Gebeurtenissen afhandelen" op pagina 133). Tijdens de vastleg- en terugkoppelfasen beweegt een gebeurtenis zich vanaf het bronweergaveobject langs bovenliggende weergaveobjecten in het weergaveoverzicht. Als een bovenliggend object zich bevindt in een andere beveiligingssandbox dan het bronweergaveobject, stopt de vastleg- en terugkoppelfase onder dat bovenliggende object, tenzij er een vertrouwensrelatie bestaat tussen de eigenaar van het bovenliggende object en die van het bronobject. Deze vertrouwensrelatie kan op de volgende manieren tot stand worden gebracht:

1. Het SWF-bestand dat eigenaar is van het bovenliggende object roept de methode `Security.allowDomain()` aan om het domein van het SWF-bestand dat eigenaar is van het bronobject toegang te verlenen.
2. Het SWF-bestand dat eigenaar is van het bronobject roept de methode `Security.allowDomain()` aan om het domein van het SWF-bestand dat eigenaar is van het bovenliggende object toegang te verlenen.

Het object LoaderInfo dat overeenkomt met een geladen bestand (en met het object Loader) bevat de volgende twee eigenschappen, die bepalend zijn voor de relatie tussen het geladen object en het object Loader: `childAllowsParent` en `parentAllowsChild`.

Voor gebeurtenissen die worden verzonden vanuit andere objecten dan weergaveobjecten worden geen beveiligingscontroles uitgevoerd en zijn ook geen andere beveiligingbeperkingen van toepassing.

Geladen media benaderen als gegevens

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voor toegang tot geladen gegevens gebruikt u de methoden `BitmapData.draw()`, `BitmapData.drawWithQuality()` en `SoundMixer.computeSpectrum()`. Standaard kunt u geen pixel- of audiogegevens verkrijgen van afbeeldingen of audio-objecten die worden gerenderd of afgespeeld door media in een andere sandbox. Met de volgende methoden kunt u echter wel toegang verlenen tot dergelijke gegevens, ook al staan de gegevens in andere sandboxen.

- In de inhoud waarmee de afbeelding wordt gerenderd of de gegevens worden afgespeeld roept u de methode `Security.allowDomain()` aan om gegevenstoegang te verlenen tot inhoud in andere domeinen.
- Voeg voor geladen afbeeldingen, geluiden of video's een URL-beleidsbestand toe op de server van het geladen bestand. In dit bestand moet toegang worden verleend aan het domein van het SWF-bestand dat via de methode `BitmapData.draw()`, `BitmapData.drawWithQuality()` of `SoundMixer.computeSpectrum()` gegevens wil extraheren uit het bestand. De methode `drawWithQuality` is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.

In de volgende secties wordt aandacht besteed aan het opvragen van bitmap-, geluids- en videogegevens.

Bitmapgegevens opvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methoden `draw()` en `drawWithQuality()` (Flash Player 11.3; AIR 3.3) van een BitmapData-object kunt u de weergegeven pixels van een willekeurig weergaveobject overnemen in het BitmapData-object. Dit kunnen de pixels zijn van een object MovieClip, een object Bitmap of een weergaveobject. Deze methoden kunnen alleen worden gebruikt voor het overnemen van pixels in het object BitmapData als aan de volgende voorwaarden wordt voldaan:

- In het geval van een bronobject dat geen geladen bitmap is, moeten het bronobject en alle onderliggende objecten (alleen voor een object Sprite of MovieClip) afkomstig zijn uit het domein van het object dat de methode draw aanroept, of ze moeten deel uitmaken van een SWF-bestand dat toegankelijk is voor de aanroeper doordat de methode `Security.allowDomain()` is aangeroepen.
- In het geval van een geladen bitmapbronobject moet het bronobject deel uitmaken van het domein van het object dat de methode draw aanroept. De bewerking is ook mogelijk als de server van het bronobject een URL-beleidsbestand bevat waarin toestemming wordt verleend aan het aanroepende domein.

Als niet aan deze voorwaarden wordt voldaan, wordt een uitzondering SecurityError gegenereerd.

Wanneer u de afbeelding laadt met de methode `load()` van de klasse Loader, kunt u een parameter `context` opgeven (een object LoaderContext). Als u de eigenschap `checkPolicyFile` van het object LoaderContext instelt op `true`, controleert Flash Player of er een bestand met URL-beleid aanwezig is op de server waarvan de afbeelding wordt geladen. Als dat het geval is en in dit bestand is vastgelegd dat het domein van het ladende SWF-bestand toegang heeft, krijgt het bestand toegang tot de gegevens in het object Bitmap. In alle andere gevallen wordt de toegang geweigerd.

U kunt ook de eigenschap `checkPolicyFile` opgeven voor een afbeelding die wordt geladen via de tag `<img>` in een tekstveld. SWF-bestanden en afbeeldingen in een tekstveld laden met de tag "SWF-bestanden en afbeeldingen in een tekstveld laden met de tag <img >" op pagina 1128

Geluidsgegevens opvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Er gelden beveiligingsbeperkingen voor de volgende API's van ActionScript 3.0 voor geluiden:

- De methode `SoundMixer.computeSpectrum()`: deze methode is altijd toegestaan voor code die wordt uitgevoerd in dezelfde beveiligingssandbox als het geluidsbestand. Voor code die in andere sandboxen wordt uitgevoerd zijn beveiligingscontroles beschikbaar.
- De methode `SoundMixer.stopAll()`: deze methode is altijd toegestaan voor code die wordt uitgevoerd in dezelfde beveiligingssandbox als het geluidsbestand. Voor bestanden in andere sandboxen zijn beveiligingscontroles beschikbaar.
- De eigenschap `id3` van de klasse Sound - Deze eigenschap is altijd toegestaan voor SWF-bestanden in dezelfde beveiligingssandbox als het geluidsbestand. Voor code die in andere sandboxen wordt uitgevoerd zijn beveiligingscontroles beschikbaar.

Elk geluid is gekoppeld aan twee soorten sandboxen: een inhoudssandbox en een eigenaarssandbox.

- Het oorspronkelijke domein van het geluid is bepalend voor de inhoudssandbox. Deze sandbox bepaalt weer of uit het geluid gegevens kunnen worden geëxtraheerd via de eigenschap `id3` van het geluid en de methode `SoundMixer.computeSpectrum()`.
- Het object dat het afspelen van het geluid heeft gestart, is bepalend voor de eigenaarssandbox. Deze sandbox bepaalt weer of het geluid kan worden gestopt via de methode `SoundMixer.stopAll()`.

Wanneer u het geluid laadt met de methode `load()` van de klasse Sound, kunt u een parameter `context` opgeven (een object SoundLoaderContext). Als u de eigenschap `checkPolicyFile` van het object SoundLoaderContext instelt op `true`, controleert de runtime of er een bestand met URL-beleid aanwezig is op de server waarvan het geluid wordt geladen. Als dat het geval is en in dit bestand is vastgelegd dat het domein van de ladende code toegang heeft, krijgt de code toegang tot de eigenschap `id` van het Sound-object. In alle andere gevallen wordt de toegang geweigerd. Via de eigenschap `checkPolicyFile` kan ook de methode `SoundMixer.computeSpectrum()` worden ingeschakeld voor geladen geluiden.

U kunt de methode `SoundMixer.areSoundsInaccessible()` gebruiken om vast te stellen of een aanroep van de methode `SoundMixer.stopAll()` tot gevolg heeft dat het afspelen van alle geluiden wordt gestopt, omdat de eigenaarssandbox van een of meer geluiden niet toegankelijk is voor de aanroeper.

Het aanroepen van de methode `SoundMixer.stopAll()` heeft tot gevolg dat het afspelen wordt gestopt van de geluiden waarvan de eigenaarssandbox dezelfde is als die van de aanroeper van `stopAll()`. De aanroep heeft eveneens tot gevolg dat het afspelen wordt gestopt van de geluiden waarvan het afspelen is gestart door SWF-bestanden die de methode `Security.allowDomain()` hebben aangeroepen om toegang te verlenen aan het domein van het SWF-bestand dat de methode `stopAll()` aanroept. Het afspelen van eventuele andere geluiden wordt niet gestopt. De aanwezigheid van deze geluiden kan worden vastgesteld door de methode `SoundMixer.areSoundsInaccessible()` aan te roepen.

De methode `computeSpectrum()` kan alleen worden aangeroepen als alle geluiden die worden afgespeeld afkomstig zijn uit de sandbox van het object dat de methode aanroept of uit een bron die toegang heeft verleend aan de sandbox van de aanroeper. Als niet aan deze voorwaarden wordt voldaan, wordt een uitzondering SecurityError gegenereerd. In het geval van geluiden die zijn geladen uit ingesloten geluiden in een bibliotheek in een SWF-bestand kan toegang worden verleend met een aanroep van de methode `Security.allowDomain()` in het geladen SWF-bestand. Voor geluiden die zijn geladen uit andere bronnen dan SWF-bestanden (zoals uit geladen MP3-bestanden of uit videobestanden) moet in een URL-beleidsbestand op de server van het bronobject toegang worden verleend tot gegevens in geladen media.

Zie "Controlemiddelen voor auteurs (ontwikkelaars)" op pagina 1120 en "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116 voor meer informatie.

Als u toegang wilt tot geluidsgegevens van RTMP-streams, moet u toegang verlenen tot de server. Met de ActionScript-eigenschap `Client.audioSampleAccess` aan de serverkant kunt u toegang verlenen tot bepaalde mappen op Flash Media Server. Zie de Server-Side ActionScript Language Reference voor meer informatie.

Videogegevens opvragen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `BitmapData.draw()` of `BitmapData.drawWithQuality()` kunt u de pixelgegevens vastleggen uit het huidige frame van een video. (De methode `drawWithQuality` is beschikbaar in Flash Player 11.3 en hoger en in AIR 3.3 en hoger.)

Er zijn twee verschillende soorten video:

- Video gestreamd over RTMP van Flash Media Server
- Progressieve video die wordt geladen van een FLV- of F4V-bestand

Als u de draw-methoden wilt gebruiken om runtimeafbeeldingen te extraheren uit RTMP-streams, moet u toegang verlenen tot de server. Met de ActionScript-eigenschap `Client.videoSampleAccess` aan de serverkant kunt u toegang verlenen tot bepaalde mappen op Flash Media Server. Zie de Server-Side ActionScript Language Reference voor meer informatie.

Wanneer u een draw-methode aanroept met progressieve video als waarde voor de parameter `source`, moet de aanroeper van de methode deel uitmaken van de sandbox van het FLV-bestand of moet op de server van het FLV-bestand een beleidsbestand bestaan waarin toegang wordt verleend aan het domein van het aanroepende SWF-bestand. U kunt aanvragen dat het beleid wordt gedownload door de eigenschap `checkPolicyFile` van het object NetStream in te stellen op `true`.

Gegevens laden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Inhoud in Flash Player en AIR kan gegevens uitwisselen met servers. Het laden van gegevens is een ander type bewerking dan het laden van media. De reden hiervoor is dat de geladen gegevens als programmaobjecten en niet als media worden weergegeven. Over het algemeen kunnen gegevens door inhoud worden geladen als de gegevens uit het domein komen waarin de inhoud oorspronkelijk is gemaakt. Gewoonlijk vereist inhoud echter beleidsbestanden om gegevens uit andere domeinen te laden (zie "Controlemiddelen voor websites (beleidsbestanden)" op pagina 1116).

Opmerking: *Inhoud die wordt uitgevoerd in de AIR-toepassingssandbox wordt nooit aangeboden vanaf een extern domein (tenzij de ontwikkelaar doelbewust externe inhoud importeert in de toepassingssandbox). Daarom kan deze inhoud ook niet deelnemen aan de typen aanvallen waartegen beleidsbestanden beveiligen. AIR-inhoud in de toepassingssandbox wordt niet ingeperkt door de beleidsbestanden bij het laden van gegevens. AIR-inhoud in andere sandboxen is echter wel onderhevig aan de beperkingen die hier worden beschreven.*

URLLoader en URLStream gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

U kunt gegevens laden, zoals een XML-bestand of een tekstbestand. Het aanroepen van de methode `load()` van de klassen URLLoader en URLStream is afhankelijk van bevoegdheden die zijn vastgelegd in een bestand met interdomeinbeleid.

Als u de methode `load()` gebruikt om inhoud te laden uit een ander domein dan het domein van de code waarmee de methode wordt aangeroepen, controleert de runtime of er op de server van de geladen elementen een URL-beleidsbestand aanwezig is. Als dat het geval is en het beleid toegang biedt tot het domein van de inhoud die wordt geladen, kunt u de gegevens laden.

Verbinding maken met sockets

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Standaard zoekt de runtime een socketbeleidsbestand dat wordt aangeleverd vanaf poort 843. Net als bij URL-beleidsbestanden wordt dit bestand het *hoofdbeleidsbestand* genoemd.

Toen beleidsbestanden voor het eerst in Flash Player 6 werden geïntroduceerd, was er geen ondersteuning voor socketbeleidsbestanden. Verbindingen met socketservers werden toegestaan door een beleidsbestand op de standaardlocatie op een HTTP-server via poort 80 van dezelfde host als de socketserver. Flash Player 9 ondersteunt deze functie nog wel, Flash Player 10 niet. In Flash Player 10 kunnen socketverbindingen alleen door socketbeleidsbestanden worden geautoriseerd.

Socketbeleidsbestanden ondersteunen, net als URL-beleidsbestanden, een metabeleidsinstructie waarin staat welke poorten beleidsbestanden kunnen leveren. In plaats van "master-only" is het standaardmetabeleid voor socketbeleidsbestanden "all". Dat wil zeggen, dat tenzij het hoofdbeleidsbestand een striktere instelling opgeeft, Flash Player ervan uitgaat dat elke socket op de host een socketbeleidsbestand kan aanleveren.

Toegang tot socket- en XML-socketverbindingen is standaard uitgeschakeld, zelfs als de socket waarmee u verbinding maakt in hetzelfde domein is als het SWF-bestand. U kunt toegang op socketniveau toestaan door een socketbeleidsbestand aan te leveren uit een van de volgende locaties:

- Poort 843 (de locatie van het hoofdbeleidsbestand)
- De poort van de hoofdsocketverbinding
- Een andere poort dan de hoofdsocketverbinding

Standaard zoekt Flash Player naar een socketbeleidsbestand op poort 843 en op dezelfde poort als de hoofdsocketverbinding. Als u een socketbeleidsbestand van een andere poort wilt aanleveren, moet het SWF-bestand `Security.loadPolicyFile()` aanroepen.

Een socketbeleidsbestand heeft dezelfde syntaxis als een URL-beleidsbestand, maar moet ook opgeven aan welke poorten toegang wordt toegewezen. Wanneer een beleidsbestand van een poortnummer lager dan 1024 afkomstig is, kan het beleidsbestand toegang verlenen tot alle andere poorten. Wanneer een beleidsbestand van poort 1024 of hoger afkomstig is, kan het beleidsbestand alleen toegang verlenen tot de poorten 1024 en hoger. De toegestane poorten worden in het attribuut `to-ports` in de tag `<allow-access-from>` opgegeven. Afzonderlijke poortnummers, poortbereiken en jokertekens zijn toegestane waarden.

Dit is een voorbeeld van een socketbeleidsbestand:

```
<?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>
```

Als u een socketbeleidsbestand van poort 843 wilt ophalen of van dezelfde poort als een hoofdsocketverbinding, roept u de methode `Socket.connect()` of `XMLSocket.connect()` aan. Flash Player controleert eerst of er een hoofdbeleidsbestand aanwezig is op poort 843. Als het er een vindt, wordt gecontroleerd of het bestand een metabeleidsinstructie bevat die socketbeleidsbestanden op de doelpoort verbiedt. Als toegang niet verboden is, zoekt Flash Player eerst naar de juiste instructie `allow-access-from` in het hoofdbeleidsbestand. Als er geen wordt gevonden, zoekt het naar een socketbeleidsbestand op dezelfde poort als de hoofdsocketverbinding.

Als u een socketbeleidsbestand van een andere locatie wilt ophalen, roept u eerst de methode `Security.loadPolicyFile()` aan met de speciale `"xmlsocket"`-syntaxis, als in het volgende voorbeeld:

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

Roep eerst de methode `Security.loadPolicyFile()` aan en vervolgens de methode `Socket.connect()` of `XMLSocket.connect()`. Flash Player wacht tot het verzoek om het beleidsbestand is afgehandeld, waarna wordt vastgesteld of de hoofdverbinding al dan niet mag worden gebruikt. Als het hoofdbeleidsbestand echter aangeeft dat de doellocatie geen beleidsbestanden mag aanleveren, heeft de aanroep naar `loadPolicyFile()` geen effect, zelfs als er op die locatie een beleidsbestand aanwezig is.

Als u bezig bent met de implementatie van een socketserver en u een bestand met socketbeleid beschikbaar wilt stellen, is het belangrijk te bepalen of u het bestand wilt aanbieden via de poort voor hoofdverbindingen of via een andere poort. In beide gevallen moet de server wachten op de eerste transmissie van de client voordat u een reactie verzendt.

Als Flash Player een beleidsbestand aanvraagt, wordt na het tot stand brengen van de verbinding altijd de volgende tekenreeks verzonden:

```
<policy-file-request/>
```

Op het moment dat de server deze tekenreeks ontvangt, kan het bestand worden verzonden. Het verzoek van Flash Player wordt altijd afgesloten met een null-byte en de reactie van de server moet ook worden afgesloten met een null-byte.

Het is niet mogelijk dezelfde verbinding te gebruiken voor het aanvragen van een beleidsbestand en als hoofdverbinding. U moet de verbinding sluiten nadat het beleidsbestand is verzonden. Als u dat niet doet, sluit Flash Player de verbinding voor de aanvraag voordat de hoofdverbinding tot stand wordt gebracht.

Gegevens beveiligen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Als u gegevens wilt beveiligen, zodat deze niet kunnen worden afgeluisterd of gewijzigd wanneer ze via internet worden verzonden, kunt u TLS (Transport Layer Security) of SSL (Socket Layer Security) toepassen op de server waar de gegevens oorspronkelijk vandaan komen. Vervolgens kunt u verbinding maken met de server via het HTTPS-protocol.

Bij toepassingen die zijn gemaakt voor AIR 2 of hoger, kunt u ook de communicatie via de TCP-socket beveiligen. Met de klasse SecureSocket kunt u een socketverbinding maken met een socketserver die gebruikmaakt van TLS versie 1 of SSL versie 4.

Gegevens verzenden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Gegevensverzending treedt op wanneer de code gegevens stuurt naar een server of resource. Gegevens kunnen altijd worden verzonden als het gaat om inhoud van een netwerkdomein. Een lokaal SWF-bestand kan alleen gegevens verzenden naar netwerkadressen als het bestand deel uitmaakt van de sandbox Lokaal-vertrouwd, Lokaal-met-netwerk of AIR-toepassing. Zie "Lokale sandboxen" op pagina 1109 voor meer informatie.

Gebruik de functie `flash.net.sendToURL()` om gegevens te verzenden naar een URL. Er zijn ook diverse methoden die aanvragen verzenden naar URL's. Voorbeelden hiervan zijn laadmethoden, zoals `Loader.load()` en `Sound.load()`, en methoden voor het laden van gegevens, zoals `URLLoader.load()` en `URLStream.load()`.

bestanden uploaden en downloaden

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de methode `FileReference.upload()` wordt het uploaden naar een externe server gestart van een bestand dat is geselecteerd door een gebruiker. U moet de methode `FileReference.browse()` of `FileReferenceList.browse()`aanroepen voordat u de methode `FileReference.upload()` aanroept.

De code die de methode `FileReference.browse()` of `FileReferenceList.browse()` initieert, kan alleen worden opgeroepen als reactie op een muis- of toetsenbordgebeurtenis. Als deze code in andere situaties wordt opgeroepen, wordt in Flash Player 10 en hoger een uitzonderingsfout gegenereerd. Voor het aanroepen van deze methoden van de AIR-toepassingssandbox is echter geen gebeurtenis vereist die door de gebruiker wordt geïnitieerd.

Het aanroepen van de methode `FileReference.download()` zorgt ervoor dat er een dialoogvenster wordt geopend waarmee de gebruiker een bestand kan downloaden van een externe server.

Opmerking: *Als op de server verificatie van gebruikers is vereist, kunnen alleen SWF-bestanden die in een browser worden uitgevoerd, met andere woorden, die de browserinsteekmodule of het ActiveX-besturingselement gebruiken, een dialoogvenster weergeven waarin de gebruiker wordt gevraagd een gebruikersnaam en wachtwoord ter verificatie op te geven (alleen voor downloads). Flash Player biedt geen ondersteuning voor uploads naar een server die gebruikersverificatie vereist.*

Uploads en downloads zijn niet toegestaan wanneer het aanroepende SWF-bestand zich in de sandbox Lokaal-met-bestandssysteem bevindt.

De standaardinstelling is dat een SWF-bestand alleen uploads en downloads kan starten naar en van de eigen server. Een SWF-bestand kan alleen gegevens uploaden naar of downloaden van een andere server als op die server een beleidsbestand beschikbaar is dat toegang biedt tot het domein van het initiërende SWF-bestand.

Ingesloten inhoud laden uit SWF-bestanden die worden geïmporteerd in een beveiligingsdomein

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer u een SWF-bestand laadt, kunt u de parameter `context` instellen van de methode `load()` van het object Loader dat wordt gebruikt om het bestand te laden. De waarde van deze parameter bestaat uit een object LoaderContext. Wanneer u de eigenschap `securityDomain` van het object LoaderContext instelt op `Security.currentDomain`, controleert Flash Player of er een bestand met interdomeinbeleid aanwezig is op de server van het geladen SWF-bestand. Als dat het geval is en het beleid toegang biedt tot het domein van het SWF-bestand dat wordt geladen, kunt u het SWF-bestand laden als geïmporteerde media. Op deze manier kan het bestand toegang krijgen tot objecten in de bibliotheek van het SWF-bestand.

Een andere manier om een SWF-bestand toegang te bieden tot klassen in geladen SWF-bestanden uit een andere beveiligingssandbox is het aanroepen van de methode `Security.allowDomain()` door het geladen SWF-bestand om toegang te verlenen aan het domein van het aanroepende SWF-bestand. U kunt de aanroep van de methode `Security.allowDomain()` toevoegen aan de constructormethode van de hoofdklasse van het geladen SWF-bestand en het ladende SWF-bestand vervolgens een gebeurtenislistener laten toevoegen om te reageren op de gebeurtenis `init` die wordt verzonden door de eigenschap `contentLoaderInfo` van het object Loader. Wanneer deze gebeurtenis wordt verzonden, heeft het geladen SWF-bestand de methode `Security.allowDomain()` in de constructormethode aangeroepen en zijn de klassen in het geladen SWF-bestand beschikbaar voor het SWF-bestand dat wordt geladen. Het ladende SWF-bestand kan klassen uit het geladen SWF-bestand ophalen door `Loader.contentLoaderInfo.applicationDomain.getDefinition()` of Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames() aan te roepen (Flash Player 11.3 en hoger; AIR 3.3 en hoger).

Werken met oude inhoud

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

In Flash Player 6 is het domein dat wordt gebruikt voor bepaalde Flash Player-instellingen gerelateerd aan het laatste gedeelte van het domein van het SWF-bestand. Het betreft hier onder andere instellingen voor camera- en microfoonbevoegdheden, opslagquota en de opslag van blijvende gezamenlijke objecten.

Als het domein van een SWF-bestand uit meer dan twee segmenten bestaat, bijvoorbeeld www.voorbeeld.com, wordt het eerste segment van het domein (www) verwijderd en wordt het resterende gedeelte van het domein gebruikt. In Flash Player 6 wordt example.com dus zowel door www.example.com als store.example.com gebruikt als het domein voor deze instellingen. Ook www.example.co.nl en store.example.co.nl gebruiken beide example.co.nl als het domein voor deze instellingen. Dit kan problemen veroorzaken wanneer SWF-bestanden uit niet-gerelateerde domeinen, zoals voorbeeld1.co.nl en voorbeeld2.co.nl, toegang hebben tot dezelfde gezamenlijke objecten.

In Flash Player 7 en hoger worden de instellingen van de speler standaard gekozen volgens het exacte domein van een SWF-bestand. Voor een SWF-bestand van www.voorbeeld.com worden bijvoorbeeld de spelerinstellingen voor www.voorbeeld.com gebruikt. Voor een SWF-bestand van winkel.voorbeeld.com worden de afzonderlijke spelerinstellingen voor winkel.voorbeeld.com gebruikt.

Als in een SWF-bestand dat is geschreven met ActionScript 3.0 de eigenschap `Security.exactSettings` is ingesteld op `true` (de standaardwaarde), gebruikt Flash Player exacte domeinen voor het bepalen van de Player-instellingen. Als de eigenschap is ingesteld op `false`, gebruikt Flash Player de domeininstellingen zoals die worden vastgesteld in Flash Player 6. Als u de standaardwaarde van `exactSettings` wijzigt, moet u dit doen voordat er gebeurtenissen plaatsvinden die vereisen dat Flash Player instellingen kiest. Voorbeelden hiervan zijn gebeurtenissen voor het gebruik van een camera of microfoon, of het opvragen van een blijvend gezamenlijk object.

Als u een SWF-bestand uit versie 6 hebt gepubliceerd en op basis hiervan blijvende gezamenlijke objecten hebt gemaakt, kunt u deze objecten opvragen uit een SWF-bestand dat ActionScript 3.0 gebruikt door `Security.exactSettings` in te stellen op `false` voordat u `SharedObject.getLocal()` aanroept.

LocalConnection-bevoegdheden instellen

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Met de LocalConnection-klasse kunt u berichten tussen verschillende Flash Player- of AIR-toepassingen verzenden. LocalConnection-objecten kunnen alleen communiceren als de Flash Player- of AIR-inhoud op dezelfde clientcomputer wordt uitgevoerd. De objecten kunnen hierbij wel in verschillende toepassingen worden uitgevoerd: zo kunnen een SWF-bestand in een browser, een SWF-bestand die in een projector wordt uitgevoerd en een AIR-toepassing alle drie communiceren met behulp van de LocalConnection-klasse.

Bij elke LocalConnection-communicatie is er een verzender en een listener. In Flash Player is standaard LocalConnection-communicatie mogelijk tussen code die wordt uitgevoerd in hetzelfde domein. Als de code wordt uitgevoerd in verschillende sandboxen, moet de listener de zender bevoegdheid verlenen via de methode `LocalConnection.allowDomain()`. De reeks die u als een argument doorgeeft aan de methode `LocalConnection.allowDomain()` kan elk van de volgende elementen bevatten: exacte domeinnamen, IP-adressen en de joker `*`.

De methode `allowDomain()` is van vorm veranderd sinds ActionScript 1.0 en 2.0. In deze eerdere versies was `allowDomain()` een callback-methode die u moest implementeren. In ActionScript 3.0 is `allowDomain()` een ingebouwde methode van de klasse LocalConnection die u aanroept. Deze wijziging betekent dat `allowDomain()` op ongeveer dezelfde manier werkt als `Security.allowDomain()`.

Een SWF-bestand kan via de eigenschap `domain` van de klasse LocalConnection het domein van de klasse bepalen.

Uitgaande URL-toegang beheren

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Uitgaande scripting en URL-toegang (met behulp van HTTP URL's, mailto:, enzovoort) kan worden bereikt met behulp van de volgende API's:

- De functie `flash.system.fscommand()`
- De methode `ExternalInterface.call()`
- De functie `flash.net.navigateToURL()`

Als de inhoud wordt geladen van het lokale bestandssysteem, kunnen deze methoden alleen succesvol worden aangeroepen als de code en de webpagina waarin de code is opgenomen (indien aanwezig) zich in de beveiligingssandboxen Lokaal-vertrouwd of AIR-toepassing bevinden. Aanroepen van deze methoden mislukken als de inhoud zich bevindt in de sandbox Lokaal-met-bestandssysteem of Lokaal-met-netwerk.

Als de inhoud niet lokaal is geladen, kunnen al deze API's communiceren met de webpagina waarin ze ingesloten zijn, afhankelijk van de waarde van de parameter AllowScriptAccess (zie hieronder voor een beschrijving). De functie `flash.net.navigateToURL()` heeft de extra optie om te communiceren met elk geopend browservenster of -frame, niet alleen met de pagina waarin het SWF-bestand is ingesloten. Zie "De functie navigateToURL() gebruiken" op pagina 1142 voor meer informatie over deze functionaliteit.

De parameter `AllowScriptAccess` in de HTML-code voor het laden van een SWF-bestand bepaalt of uitgaande scripts mogelijk zijn vanuit een SWF-bestand. Stel deze parameter in de tag PARAM of EMBED in. Als er geen waarde is ingesteld voor `AllowScriptAccess`, kunnen het SWF-bestand en de HTML-pagin alleen communiceren als ze van hetzelfde domein afkomstig zijn.

De parameter `AllowScriptAccess` kan een van de volgende drie mogelijke waarden hebben: `"always"`, `"sameDomain"` of `"never"`.

- Wanneer `AllowScriptAccess` is ingesteld op `"always"`, kan het SWF-bestand communiceren met de HTML-pagina waarin het is ingesloten, zelfs als het SWF-bestand afkomstig is van een ander domein dan de HTML-pagina.
- Wanneer `AllowScriptAccess` is ingesteld op `"sameDomain"`, kan het SWF-bestand alleen communiceren met de HTML-pagina waarin het is ingesloten als het SWF-bestand afkomstig is van hetzelfde domein als de HTML-pagina. Deze waarde is de standaardwaarde voor `AllowScriptAccess`. Gebruik deze instelling of stel geen waarde in voor `AllowScriptAccess` om te voorkomen dat een SWF-bestand dat op het ene domein wordt aangeboden, toegang krijgt tot een script in een HTML-pagina die uit een ander domein afkomstig is.
- Wanneer `allowScriptAccess` is ingesteld op `"never"`, kan het SWF-bestand niet communiceren met een HTML-pagina. Het gebruik van deze waarde is afgekeurd sinds de release van Adobe Flash CS4 Professional. Het gebruik ervan wordt niet aanbevolen en is ook niet nodig als u vanuit uw domein niet met niet-vertrouwde SWF-bestanden werkt. Als u wel niet-vertrouwde SWF-bestanden beschikbaar moet stellen, raadt Adobe u aan een afzonderlijk subdomein te maken en alle niet-vertrouwde inhoud daar te plaatsen.

Hier is een voorbeeld van het instellen van de tag `AllowScriptAccess` in een HTML-pagina zodat uitgaande URL-toegang naar een ander domein wordt toegestaan:

```
<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>
```

De functie navigateToURL() gebruiken

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Naast de beveiligingsinstelling die is opgegeven met de parameter `allowScriptAccess` die hierboven wordt beschreven, heeft de functie `navigateToURL()` een optionele tweede parameter: `target`. De parameter `target` kan worden gebruikt om de naam van een HTML-venster of -frame op te geven waarnaar het URL-verzoek moet worden verzonden. Voor zulke verzoeken gelden extra beveiligingsbeperkingen en deze hangen af van de vraag of `navigateToURL()` wordt gebruikt als een scripting- of niet-scriptinginstructie.

Voor scriptinginstructies, zoals `navigateToURL("javascript: alert('Hello from Flash Player.')")` gelden de volgende regels.

- Als het SWF-bestand een lokaal vertrouwd bestand is, wordt het verzoek gehonoreerd.
- Als het doel de HTML-pagina is waarin het SWF-bestand is ingesloten, gelden de `allowScriptAccess`-regels die hierboven zijn beschreven.
- Als het doel inhoud bevat die is geladen van hetzelfde domein als het SWF-bestand, wordt het verzoek gehonoreerd.
- Als het doel inhoud bevat die is geladen uit een ander domein dan het SWF-bestand en als aan geen van de eerder genoemde twee voorwaarden wordt voldaan, wordt de aanvraag niet gehonoreerd.

Voor niet-scriptinginstructies (zoals HTTP, HTTPS en `mailto:` wordt de aanvraag niet gehonoreerd als aan alle volgende voorwaarden wordt voldaan:

- Het doel is een van de speciale trefwoorden `"_top"` of `"_parent"` en
- het SWF-bestand bevindt zich in een webpagina die afkomstig is uit een ander domein en
- het SWF-bestand is ingesloten met een waarde voor `allowScriptAccess` die niet `"always"` is.

Voor meer informatie

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Voor meer informatie over uitgaande URL-toegang gaat u naar de volgende gedeelten in de *Naslaggids voor ActionScript 3.0 voor het Adobe Flash-platform:*

- De flash.system.fscommand()-functie
- De call()-methode van de ExternalInterface-klasse
- De `flash.net.navigateToURL()`-functie

Gezamenlijke objecten

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Flash Player biedt ondersteuning voor het werken met *gezamenlijke objecten*. Het betreft hier ActionScript-objecten die beschikbaar blijven buiten een SWF-bestand, hetzij lokaal in het bestandssysteem van een gebruiker of extern op een RTMP-server. Gezamenlijke objecten worden net als andere media in Flash Player ingedeeld in beveiligingssandboxen. Het sandboxmodel voor gezamenlijke objecten wijkt echter wel iets af, aangezien gezamenlijke objecten geen bronnen zijn die vanuit een extern domein kunnen worden benaderd. Gezamenlijke objecten worden

daarentegen altijd opgehaald van een speciale opslaglocatie die is ingesteld voor het domein van het SWF-bestand dat methoden van de klasse SharedObject aanroept. Normaal gesproken is de opslaglocatie van een gezamenlijk object nog specifieker dan het domein van een SWF-bestand: elk SWF-bestand gebruikt standaard de opslaglocatie voor gezamenlijke objecten die specifiek is voor de gehele oorspronkelijke URL. Zie "Gezamenlijke objecten" op pagina 741 voor meer informatie over gezamenlijke objecten.

Een SWF-bestand kan via de parameter `localPath` van de methoden `SharedObject.getLocal()` en `SharedObject.getRemote()` een opslaglocatie van gezamenlijke objecten gebruiken die aan slechts een deel van de URL is gekoppeld. Op deze manier kan het SWF-bestand de gezamenlijke objecten ook delen met andere SWF-bestanden uit andere URL's. Zelfs als u de waarde `'/'` opgeeft voor de parameter `localPath`, wordt er een opslaglocatie van gezamenlijke objecten gebruikt die specifiek is voor het eigen domein.

Gebruikers kunnen de toegang tot gezamenlijke objecten beperken via het dialoogvenster Flash Player Settings of via Settings Manager. Standaard kunnen gezamenlijke objecten per domein maximaal 100 kB aan gegevens bevatten. Beheerders en gebruikers kunnen ook beperkingen instellen voor het schrijven naar het bestandssysteem. Zie "Controlemiddelen voor beheerders" op pagina 1113en "Controlemiddelen voor gebruikers" op pagina 1115 voor meer informatie.

U kunt instellen dat een gezamenlijk object beveiligd is door `true` op te geven voor de parameter `secure` van de methode `SharedObject.getLocal()` of `SharedObject.getRemote()`. De volgende punten zijn van belang voor de parameter `secure`:

- Als deze parameter wordt ingesteld op `true`, wordt er in Flash Player een nieuw beveiligd gezamenlijk object gemaakt of wordt er een verwijzing naar een bestaand beveiligd gezamenlijk object opgehaald. Alleen SWF-bestanden die via HTTPS worden geleverd en `SharedObject.getLocal()` aanroepen met de parameter secure ingesteld op `true` hebben lees- en schrijftoegang tot dit beveiligde gezamenlijke object.
- Als deze parameter wordt ingesteld op `false`, wordt er in Flash Player een nieuw gezamenlijk object gemaakt of wordt er een verwijzing opgehaald naar een bestaand gezamenlijk object, waartoe SWF-bestanden die via niet-HTTPS-verbindingen worden geleverd, lees- en schrijftoegang hebben.

Als het aanroepende SWF-bestand niet afkomstig is van een HTTPS-URL, heeft het opgeven van `true` voor de parameter `secure` van de methode `SharedObject.getLocal()` of `SharedObject.getRemote()` een uitzondering SecurityError tot gevolg.

De keuze van een opslaglocatie van gezamenlijke objecten is afhankelijk van de oorspronkelijke URL van een SWF-bestand. Dit is zelfs het geval in de twee situaties waarin een SWF-bestand niet afkomstig is van een eenvoudige URL: importladen en dynamisch laden. Importladen heeft betrekking op de situatie waarin een SWF-bestand wordt geladen terwijl de eigenschap `LoaderContext.securityDomain` van het bestand is ingesteld op `SecurityDomain.currentDomain`. In deze situatie heeft het geladen SWF-bestand een pseudo-URL die begint met het domein van het ladende SWF-bestand en verder bestaat uit de werkelijke oorspronkelijke URL. Dynamisch laden is het laden van een SWF-bestand met de methode `Loader.loadBytes()`. In deze situatie heeft het geladen SWF-bestand een pseudo-URL die begint met de volledige URL van het ladende SWF-bestand, gevolgd door een id die uit een geheel getal bestaat. Bij zowel importladen als dynamisch laden kan de pseudo-URL van een SWF-bestand worden gecontroleerd met de eigenschap `LoaderInfo.url`. De pseudo-URL wordt bij het kiezen van een opslaglocatie van gezamenlijke objecten op exact dezelfde manier verwerkt als een echte URL. U kunt voor een gezamenlijk object een waarde opgeven voor de parameter `localPath` waarin de pseudo-URL geheel of gedeeltelijk wordt gebruikt.

Gebruikers en beheerders kunnen desgewenst het gebruik van *gezamenlijke objecten van derden* uitschakelen. Het betreft hier het gebruik van gezamenlijke objecten door een SWF-bestand dat wordt uitgevoerd in een webbrowser en waarbij de oorspronkelijke URL van dat SWF-bestand afkomstig is uit een ander domein dan de URL in de adresbalk van de browser. Gebruikers en beheerders kunnen het gebruik van gezamenlijke objecten van derden uitschakelen vanwege privacyredenen. Het is dan namelijk niet mogelijk objecten te traceren buiten het eigen domein. Deze beperking is niet nodig als u ervoor zorgt dat een SWF-bestand dat gezamenlijke objecten gebruikt, alleen wordt

geladen binnen HTML-paginastructuren die ervoor zorgen dat het SWF-bestand afkomstig is uit het domein dat wordt weergegeven in de adresbalk van de browser. Wanneer u probeert gezamenlijke objecten te gebruiken uit een SWF-bestand van derden en het gebruik van gezamenlijke objecten van derden is uitgeschakeld, resulteren de methoden `SharedObject.getLocal()` en `SharedObject.getRemote()` in `null`. Zie www.adobe.com/products/flashplayer/articles/thirdpartylso voor meer informatie.

Toegang tot camera, microfoon, klembord, muis en toetsenbord

Flash Player 9 of hoger, Adobe AIR 1.0 of hoger

Wanneer een SWF-bestand via de methode `Camera.get()` of `Microphone.get()` toegang probeert te krijgen tot de camera of microfoon van een gebruiker, wordt in Flash Player een dialoogvenster Privacy weergegeven waarin de gebruiker de toegang tot de camera en microfoon kan toestaan of weigeren. De gebruiker en de gebruiker met administratieve bevoegdheden kan de toegang tot camera's ook per site of globaal verbieden. Hiervoor zijn instellingen beschikbaar in het bestand mms.cfg, de Settings-gebruikersinterface en Instellingenbeheer(zie "Controlemiddelen voor beheerders" op pagina 1113 en "Controlemiddelen voor gebruikers" op pagina 1115). Als de gebruiker beperkingen heeft ingesteld voor toegang tot de camera en microfoon, resulteren de methoden `Camera.get()` en `Microphone.get()` beide in `null`. Met de eigenschap `Capabilities.avHardwareDisable` kunt u vaststellen of de camera en microfoon toegankelijk zijn (`false`) of worden geblokkeerd (`true`).

Via de methode `System.setClipboard()` kan een SWF-bestand de inhoud van het klembord vervangen door een tekenreeks met onbewerkte tekst. Hierdoor ontstaat geen beveiligingsrisico. Het risico dat zou kunnen ontstaan door het knippen of kopiëren van wachtwoorden en andere vertrouwelijke gegevens naar het klembord wordt namelijk voorkomen doordat er geen overeenkomstige methode `getClipboard` (inhoud van klembord lezen) is.

Een toepassing die in Flash Player wordt uitgevoerd, kan alleen toetsenbord- of muisgebeurtenissen bewaken die binnen de focus van die toepassing worden uitgevoerd. Inhoud die in Flash Player wordt uitgevoerd, kan geen toetsenbord- of muisgebeurtenissen in andere toepassingen detecteren.

Beveiliging in AIR

Adobe AIR 1.0 of hoger

Basisbeginselen van beveiliging in AIR

Adobe AIR 1.0 of hoger

AIR-toepassingen worden met dezelfde beveiligingsbeperkingen als native toepassingen uitgevoerd. Over het algemeen bieden AIR-toepassingen net als native toepassingen toegang tot functies van het besturingssysteem, zoals het lezen en schrijven van bestanden, het starten van toepassingen, het tekenen op het scherm en het communiceren met het netwerk. Beperkingen van het besturingssysteem die gelden voor native toepassingen, zoals gebruikersspecifieke rechten, gelden ook voor AIR-toepassingen.

Hoewel het beveiligingsmodel van Adobe® AIR® geëvolueerd is uit het beveiligingsmodel van Adobe® Flash® Player, verschilt het beveiligingscontract van het contract dat op de inhoud van een browser wordt toegepast. Dit contract biedt ontwikkelaars een veilige manier om te beschikken over een bredere functionaliteit voor interessante ervaringen, met een vrijheid die niet geschikt is voor browsertoepassingen.

AIR-toepassingen worden geschreven in gecompileerde bytecode (SWF-inhoud) of met geïnterpreteerde scripts (JavaScript, HTML), zodat de runtime geheugenbeheer biedt. Dit beperkt de kwetsbaarheid van AIR-toepassingen voor geheugenbeheerproblemen zoals buffer overflow en gegevensbeschadiging. Hieronder ziet u een aantal van de meest voorkomende problemen bij desktoptoepassingen die in native code zijn geschreven.

Installatie en updates

Adobe AIR 1.0 of hoger

AIR-toepassingen worden gedistribueerd via AIR-installatiebestanden met de bestandstoevoeging `air`of via native installatieprogramma's die de bestandsindeling en extensie van het native platform gebruiken. De native indeling van Windows-installatiebestanden is bijvoorbeeld een EXE-bestand. Voor Android is het een APK-bestand.

Wanneer Adobe AIR wordt geïnstalleerd en een AIR-installatiebestand wordt geopend, beheert de AIR-runtime de installatiemethode. Wanneer een native installatieprogramma wordt gebruikt, beheert het besturingssysteem het installatieproces.

Opmerking: *Ontwikkelaars kunnen een versie, toepassingsnaam en uitgeversbron opgeven wanneer de AIR-bestandsindeling wordt gebruikt, maar de initiële workflow zelf van de toepassingsinstallatie kan niet worden gewijzigd. Deze beperking is nuttig voor de gebruiker omdat hierdoor alle AIR-toepassingen dezelfde veilige, gestroomlijnde en consistente installatiemethode hebben, die door de runtime wordt beheerd. Als een toepassing moet worden aangepast, kan dit plaatsvinden wanneer de toepassing de eerste keer wordt gestart.*

Installatielocatie van runtime

Adobe AIR 1.0 of hoger

Voor AIR-toepassingen die gebruikmaken van de AIR-bestandsindeling, moet eerst de runtime op de computer van de gebruiker worden geïnstalleerd, zoals voor SWF-bestanden eerst de Flash Player plug-in voor de browser moet worden geïnstalleerd.

De runtime wordt op de volgende locatie op pc's geïnstalleerd:

- Mac OS: `/Library/Frameworks/`
- Windows: `C:\Program Files\Common Files\Adobe AIR`
- Linux: `/opt/Adobe AIR/`

Als een gebruiker in Mac OS een recentere versie van een toepassing wil installeren, moet de gebruiker over voldoende toegangsrechten beschikken om te installeren in de toepassingsmap. In Windows en Linux moet de gebruiker over beheerdersrechten beschikken.

Opmerking: *Op iOS wordt de AIR-runtime niet afzonderlijk geïnstalleerd. Elke AIR-toepassing is een zelfstandige toepassing.*

De runtime kan op twee manieren worden geïnstalleerd: met behulp van de naadloze installatiefunctie (installatie rechtstreeks vanuit een webbrowser) of via een handmatige installatie. AIR-toepassingen die zijn verpakt als native installatieprogramma's kunnen de AIR-runtime ook installeren als onderdeel van het installatieproces van de toepassing. (Als u de AIR-runtime op deze manier distribueert is een redistributie-overeenkomst met Adobe vereist.)

Naadloze installatie (runtime en toepassing)

Adobe AIR 1.0 of hoger

De naadloze installatiefunctie biedt ontwikkelaars een gestroomlijnde installatiemethode voor gebruikers die Adobe AIR nog niet hebben geïnstalleerd. Bij de naadloze installatiemethode maakt de ontwikkelaar een SWF-bestand dat de te installeren toepassing opgeeft. Wanneer de gebruiker op het SWF-bestand klikt om de toepassing te installeren, probeert het SWF-bestand de runtime te detecteren. Als de runtime niet kan worden gedetecteerd, wordt deze geïnstalleerd en onmiddellijk gestart met de installatiemethode voor de toepassing van de ontwikkelaar.

Handmatige installatie

Adobe AIR 1.0 of hoger

De gebruiker kan echter ook de runtime handmatig downloaden en installeren vooraleer een AIR-bestand te openen. In dat geval beschikt de ontwikkelaar over verschillende manieren om een AIR-bestand te distribueren (bijvoorbeeld via e-mail of een HTML-koppeling op een website). Nadat het AIR-bestand is geopend, begint de runtime de toepassing te installeren.

Installatieflow van toepassing

Adobe AIR 1.0 of hoger

Het beveiligingsmodel van AIR biedt de gebruiker de keuze om een AIR-toepassing al dan niet te installeren. De installatiemethode van AIR biedt meerdere voordelen ten opzichte van native installatietechnologieën voor toepassingen, wat het maken van dit vertrouwensbesluit vergemakkelijkt voor de gebruiker:

- De runtime biedt een consistente installatiemethode in alle besturingssystemen, zelfs als een AIR-toepassing via een koppeling in een webbrowser wordt geïnstalleerd. De meeste native installatiemethoden voor toepassingen zijn afhankelijk van de browser of een andere toepassing voor het leveren van beveiligingsinformatie en in sommige gevallen wordt zelfs helemaal geen beveiligingsinformatie geleverd.
- De AIR-installatiemethode voor toepassingen identificeert de bron van de toepassing en informatie over de toegangsrechten van de toepassing (mits de gebruiker de installatie laat plaatsvinden).
- De runtime beheert de installatiemethode van een AIR-toepassing. Een AIR-toepassing heeft geen controle over de installatiemethode die door de runtime wordt gebruikt.

De gebruiker wordt doorgaans niet aangeraden desktoptoepassingen te installeren die afkomstig zijn van een niet-vertrouwde bron of een bron die niet kan worden gecontroleerd. De bewijslast op het gebied van veiligheid bij native toepassingen geldt ook voor AIR-toepassing en andere installeerbare toepassingen.

Doellocatie van toepassing

Adobe AIR 1.0 of hoger

De installatiedirectory kan worden ingesteld met een van de volgende twee opties:

1. De gebruiker kiest de doellocatie tijdens de installatie. De toepassing wordt geïnstalleerd op de locatie die door de gebruiker is opgegeven.
2. Als de gebruiker de standaard installatielocatie niet wijzigt, wordt de toepassing geïnstalleerd volgens het standaardpad dat door de runtime wordt bepaald:
   - Mac OS: `~/Applications/`
   - Windows XP en oudere versies: `C:\Program Files\`

- Windows Vista: `~/Apps/`
- Linux: /opt/

Als de ontwikkelaar in het descriptorbestand van de toepassing een waarde opgeeft voor `installFolder`, wordt de toepassing geïnstalleerd in een subpad van deze directory.

Bestandssysteem van AIR

Adobe AIR 1.0 of hoger

Tijdens de installatie van een AIR-toepassing worden alle bestanden die de ontwikkelaar in het installatiebestand van de AIR-toepassing heeft opgenomen, gekopieerd naar de lokale computer van de gebruiker. De geïnstalleerde toepassing bestaat uit:

- Windows: een directory met alle bestanden die in het installatiebestand van de AIR-toepassing zijn opgenomen. Tijdens de installatie van de AIR-toepassing maakt de runtime ook een EXE-bestand.
- Linux: Een map met alle bestanden die in het AIR-installatiebestand zijn opgenomen. De runtime maakt ook een BIN-bestand tijdens de installatie van de AIR-toepassing.
- Mac OS: een `APP`-bestand met de volledige inhoud van het installatiebestand van de AIR-toepassing. Dit kan worden geïnspecteerd met de opdracht Toon pakketinhoud in Finder. Tijdens de installatie van de AIR-toepassing maakt de runtime dit APP-bestand.

AIR-toepassingen worden als volgt gestart:

- Windows: voer het bestand met de extensie .exe in de installatiemap of een snelkoppeling naar dit bestand uit (bijvoorbeeld in het menu Start of op het bureaublad).
- Linux: Het BIN-bestand starten in de installatiemap, de toepassing kiezen in het menu Toepassingen of uitvoeren vanuit een alias of koppeling op het bureaublad.
- Mac OS: voer het bestand met de extensie .app of een alias van dit bestand uit.

Het bestandssysteem van de toepassing bevat ook subdirectory's betreffende de functie van de toepassing. Bijvoorbeeld: informatie die naar een gecodeerde lokale opslaglocatie wordt geschreven, wordt opgeslagen in een subdirectory in een directory met de naam van de toepassingsidentificatie.

Opslag van AIR-toepassing

Adobe AIR 1.0 of hoger

AIR-toepassingen hebben de vereiste rechten om te schrijven naar een willekeurige locatie op de vaste schijf van de gebruiker. Ontwikkelaars wordt echter aangeraden om het pad `app-storage:/` voor lokale opslag van hun toepassing te gebruiken. Bestanden die vanuit een toepassing naar `app-storage:/` worden geschreven, bevinden zich op een standaardlocatie die afhankelijk is van het besturingssysteem van de gebruiker:

- In Mac OS: de opslagdirectory van een toepassing varieert voor elke AIR-versie:
  - **AIR 3.2 en lager** - `<toepassingsgegevens>/<toepassings-id>/Local Store/` waarbij `<toepassingsgegevens>` de voorkeursmap van de gebruiker is, meestal: `/Users/<gebruiker>/Library/Preferences`
  - **AIR 3.3 en hoger** - `<pad>/Library/Application Support/<toepassings-id>/Local Store`, waarbij `<pad>` ofwel `/Users/<gebruiker>/Library/Containers/<bundle-id>/Data` is (sandbox-omgeving), of `/Users/<gebruiker>` (buiten een sandbox-omgeving)

- In Windows: de opslagmap van een toepassing is `<appData>\<appId>\Local Store\`, waarbij `<appData>` de CSIDL_APPDATA 'speciale map' van de gebruiker is, meestal: `C:\Documents and Settings\< gebruiker>\Application Data`
- In Linux: `<appData>/<appID>/Local Store/`, waarbij `<appData>/home/< gebruiker>/.appdata` is

U hebt toegang tot de opslagmap voor toepassingen via de eigenschap `air.File.applicationStorageDirectory`. U kunt de inhoud ervan bekijken via de methode `resolvePath()` van de klasse File. Zie "Werken met het bestandssysteem" op pagina 689 voor meer informatie.

Adobe AIR updaten

Adobe AIR 1.0 of hoger

Wanneer de gebruiker een AIR-toepassing installeert waarvoor een recentere runtime is vereist, installeert de runtime automatisch de vereiste update.

Voor het bijwerken van de runtime moet de gebruiker over beheerdersrechten voor de computer beschikken.

AIR-toepassingen bijwerken

Adobe AIR 1.0 of hoger

De ontwikkeling en distributie van software-updates is een van de grootste beveiligingsproblemen bij toepassingen in native code. De API van AIR biedt een mechanisme om dit risico te beperken: bij het starten kan de methode `Updater.update()` worden opgeroepen om een AIR-bestand te zoeken in een externe locatie. Als een update wordt gevonden, wordt het AIR-bestand gedownload en geïnstalleerd, en wordt de toepassing opnieuw gestart. Ontwikkelaars kunnen deze klasse niet alleen voor nieuwe functionaliteit gebruiken maar ook voor het bieden van oplossingen voor mogelijke beveiligingsproblemen.

De Updater-klasse kan alleen worden gebruikt voor het bijwerken van toepassingen die als AIR-bestanden worden gedistribueerd. Toepassingen die als native toepassingen worden gedistribueerd, dienen de eventuele updatefuncitonaliteit van het native besturingssysteem te gebruiken.

Opmerking: *Ontwikkelaars kunnen de versie van een toepassing opgeven door de eigenschap versionNumber van het descriptorbestand van de toepassing in te stellen.*

AIR-toepassingen verwijderen

Adobe AIR 1.0 of hoger

Bij het verwijderen van een AIR-toepassing worden alle bestanden uit de toepassingsmap verwijderd. Niet alle bestanden die de toepassing buiten de toepassingsmap heeft gemaakt, worden echter verwijderd. Bij het verwijderen van een AIR-toepassing worden eventuele wijzigingen die de AIR-toepassing heeft aangebracht in bestanden buiten de toepassingsmap, niet ongedaan gemaakt.

Windows-registerinstellingen voor beheerders

Adobe AIR 1.0 of hoger

In Windows kunnen beheerders een computer zo configureren dat de installatie van AIR-toepassingen en het uitvoeren van runtime-updates (niet) is toegestaan. Deze instellingen staan in het Windows-register onder de sleutel HKLM\Software\Policies\Adobe\AIR. De instellingen zijn onder andere:

Registerinstelling	Beschrijving
AppInstallDisabled	Geeft aan of het installeren en verwijderen van AIR-toepassingen is toegestaan. Zet op 0 voor 'toegestaan' of op 1 voor 'niet toegestaan'.
UntrustedAppInstallDisabled	Geeft aan dat het installeren van niet-vertrouwde AIR-toepassingen (toepassingen zonder een vertrouwd certificaat) is toegestaan. Zet op 0 voor 'toegestaan' of op 1 voor 'niet toegestaan'.
UpdateDisabled	Geeft aan of het updaten van de runtime is toegestaan, als achtergrondtaak of als deel van een expliciete installatie. Zet op 0 voor 'toegestaan' of op 1 voor 'niet toegestaan'.

HTML-beveiliging in Adobe AIR

Adobe AIR 1.0 of hoger

In dit onderwerp wordt de AIR HTML-beveiligingsarchitectuur beschreven en leest u hoe iframes, frames en de sandboxbridge worden gebruikt voor het instellen van op HTML gebaseerde toepassingen en hoe u HTML-inhoud veilig kunt integreren in op SWF gebaseerde toepassingen.

De runtime past regels toe en biedt mechanismen voor het verhelpen van mogelijke beveiligingsproblemen in HTML en JavaScript. Hierbij worden altijd dezelfde regels toegepast, ongeacht of uw toepassing hoofdzakelijk in JavaScript is geschreven of dat u de HTML- en JavaScript-inhoud in een SWF-toepassing laadt. Inhoud in de sandbox van een toepassing en inhoud in de sandbox die niet van een toepassing is, hebben verschillende rechten. Wanneer inhoud in een iframe of frame wordt geladen, biedt de runtime een veilig *sandboxbridge*-mechanisme, waardoor de inhoud van het frame of iframe veilig kan communiceren met de inhoud van de toepassingsbeveiligingssandbox.

De AIR SDK bevat drie klassen voor het renderen van HTML-inhoud.

De HTMLLoader-klasse verschaft nauwe integratie tussen JavaScript-code en de AIR API's.

De StageWebView-klasse is een HTML-renderingklasse die slechts in zeer beperkte mate integreert met de AIR-hosttoepassing. Door de StageWebView-klasse geladen inhoud wordt nooit in de beveiligingssandbox van de toepassing geplaatst en kan geen gegevens benaderen of functies aanroepen in de AIR-hosttoepassing. Op bureaubladplatformen gebruikt de StageWebView-klasse de geïntegreerde, op Webkit gebaseerde AIR HTML-engine die ook door de HTMLLoader-klasse wordt gebruikt. Op mobiele platformen gebruikt de StageWebView-klasse het door het besturingssysteem verschafte HTML-controlemiddel. Op mobiele platformen gelden voor de StageWebView-klasse dus dezelfde beveiligingsoverwegingen en en -problemen als voor de webbrowser van het systeem.

De TextField-klasse kan tekenreeksen met HTML-tekst weergeven. Er kan geen JavaScript worden uitgevoerd, maar de tekst kan koppelingen en extern geladen afbeeldingen bevatten.

Zie “JavaScript-beveiligingsfouten voorkomen” op pagina 1041voor meer informatie.

Overzicht van het configureren van uw HTML-toepassing

Adobe AIR 1.0 of hoger

Frames en iframes bieden een handige structuur voor het indelen van HTML-inhoud in AIR. Met frames kunt u zowel gegevens persistent houden als veilig werken met externe inhoud.

Aangezien HTML in AIR zijn normale, op pagina's gebaseerde indeling behoudt, wordt de HTML-omgeving volledig vernieuwd als het bovenste frame van de HTML-inhoud naar een andere pagina 'navigeert'. U kunt frames en iframes gebruiken om gegevenspersistentie in AIR te behouden, ongeveer op dezelfde manier als voor een webtoepassing in een browser. Als u de belangrijkste toepassingsobjecten in het bovenste frame plaatst, persisteren ze tot u het frame naar een nieuwe pagina laat navigeren. Gebruik subframes of -iframes om de overgangsdelen van de toepassing te laden en weer te geven. (Er zijn verschillende manieren om voor gegevenspersistentie te zorgen als aanvulling op of in plaats van frames. Bijvoorbeeld cookies, lokale gedeelde objecten, lokale bestandsopslag, gecodeerde bestandsopslag en lokale databaseopslag.)

Aangezien HTML in AIR ook de normale vage grens tussen uitvoerbare code en gegevens heeft, plaatst AIR inhoud in het bovenste frame van de HTML-omgeving in de toepassingssandbox. Na de gebeurtenis `load` waarbij de pagina wordt geladen, beperkt AIR alle bewerkingen zoals `eval()` waarbij een tekenreeks kan worden geconverteerd naar een uitvoerbaar object. Deze beperking wordt altijd toegepast, zelfs als een toepassing geen externe inhoud laadt. Als HTML-inhoud deze beperkte bewerkingen mag uitvoeren, moet u frames of iframes gebruiken om de inhoud in een niet-toepassingssandbox te plaatsen. (Soms moet inhoud in een onderliggend sandboxframe worden uitgevoerd wanneer wordt gewerkt met JavaScript-toepassingsraamwerken die zijn gebaseerd op de functie `eval()`.) Zie "Codebeperkingen voor de inhoud van verschillende sandboxen" op pagina 1152voor de volledige lijst van JavaScript-beperkingen in de toepassingssandbox.

Aangezien HTML in AIR de mogelijkheid behoudt om externe, mogelijk onveilige inhoud te laden, past AIR een 'beleid van zelfde oorsprong' toe om de voorkomen dat de inhoud van een bepaald domein in contact komt met de inhoud van een ander domein. Als u contact tussen de inhoud van de toepassing en de inhoud van een ander domein wilt toestaan, kunt u een bridge creëren die fungeert als interface tussen een hoofd- en een subframe.

Relatie tussen hoofd- en subsandbox creëren

Adobe AIR 1.0 of hoger

AIR voegt de kenmerken `sandboxRoot` en `documentRoot` toe aan de HTML frame- en iframe-elementen. Met deze kenmerken kunt u de toepassingsinhoud behandelen alsof deze van een ander domein afkomstig is:

Attribuut	Beschrijving
sandboxRoot	De URL waarmee de sandbox en het domein worden bepaald waarin de frame-inhoud wordt geplaatst. Het URL-schema `file:`, `http:` of `https:` moet worden gebruikt.
documentRoot	De URL waarvan de frame-inhoud moet worden geladen. Het URL-schema `file:`, `app:` of `app-storage:` moet worden gebruikt.

In het volgende voorbeeld wordt de inhoud van de sandboxsubdirectory van de toepassing zo geconfigureerd dat deze wordt uitgevoerd in de externe sandbox en het domein www.example.com:

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Bridge tussen hoofd- en een subframes in verschillende sandboxen of domeinen instellen

Adobe AIR 1.0 of hoger

AIR voegt de eigenschappen `childSandboxBridge` en `parentSandboxBridge` toe aan het `window`-object van een willekeurig subframe. Met deze eigenschappen kunt u bridges definiëren als interfaces tussen een hoofd- en een subframe. Elke bridge gaat in één richting:

`childSandboxBridge` - met de eigenschap `childSandboxBridge` kan het subframe een interface toegankelijk maken voor de inhoud van het hoofdframe. Als u een interface toegankelijk wilt maken, stelt u de eigenschap `childSandbox` in op een functie of object in het subframe. Vervolgens hebt u toegang tot het object of de functie vanuit de inhoud van het hoofdframe. In het volgende voorbeeld ziet u hoe een script dat in een subframe wordt uitgevoerd, een object dat een functie en een eigenschap bevat, toegankelijk kan maken voor het hoofdframe:

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Als deze onderliggende inhoud zich bevindt in een iframe waaraan de `id"child"` is toegewezen, hebt u vanuit de bovenliggende inhoud toegang tot de interface door de eigenschap `childSandboxBridge` van het frame te lezen:

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

`parentSandboxBridge` - met de eigenschap `parentSandboxBridge` kan het bovenliggende frame een interface toegankelijk maken voor de inhoud van het subframe. Als u een interface toegankelijk wilt maken, stelt u de eigenschap `parentSandbox` van het onderliggende frame in op een functie of object in het bovenliggende frame. Vervolgens hebt u toegang tot het object of de functie vanuit de inhoud van het onderliggende frame. In het volgende voorbeeld ziet u hoe een script dat in het bovenliggende frame wordt uitgevoerd, een object dat de functie Save bevat, toegankelijk kan maken voor een onderliggend frame:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

Via deze interface kan de inhoud van het onderliggende frame bijvoorbeeld tekst opslaan in een bestand met de naam save.txt. De inhoud heeft echter geen andere toegang tot het bestandssysteem. Normaliter moet de toepassingsinhoud de kleinst mogelijke interface toegankelijk maken voor andere sandboxen. De onderliggende inhoud kan bijvoorbeeld de functie Save als volgt aanroepen:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Als onderliggende inhoud een eigenschap van het object `parentSandboxBridge` probeert in te stellen, genereert de runtime een SecurityError-fout. Als bovenliggende inhoud een eigenschap van het object `childSandboxBridge` probeert in te stellen, genereert de runtime een SecurityError-fout.

Codebeperkingen voor de inhoud van verschillende sandboxen

Adobe AIR 1.0 of hoger

Zoals beschreven in de inleiding van dit onderwerp, "HTML-beveiliging in Adobe AIR" op pagina 1149, dwingt de runtime regels af en biedt deze mechanismen voor het overwinnen van mogelijke zwakheden in de beveiliging in HTML en JavaScript. De desbetreffende beperkingen staan in dit onderwerp. Als code deze beperkte API's probeert op te roepen, geeft de runtime de foutmelding weer dat er in de runtime van Adobe AIR een beveiligingsfout betreffende JavaScript-code in de toepassingsbeveiligingssandbox is opgetreden.

Zie "JavaScript-beveiligingsfouten voorkomen" op pagina 1041voor meer informatie.

Beperkingen betreffende het gebruik van de JavaScript-functie eval() en soortgelijke technieken

Adobe AIR 1.0 of hoger

Voor de HTML-inhoud van de toepassingsbeveiligingssandbox zijn er beperkingen betreffende het gebruik van API's die tekenreeksen dynamisch in programmacode kunnen omzetten nadat de code is geladen (nadat de gebeurtenis `onload` van het element `body` is verzonden en de handlerfunctie `onload` is voltooid). Hierdoor wordt voorkomen dat de toepassing onbedoeld code van niet-toepassingsbronnen (zoals mogelijk onveilige netwerkdomeinen) opneemt (en uitvoert).

Als uw toepassing bijvoorbeeld tekenreeksgegevens van een externe bron gebruikt om naar de eigenschap innerHTML van een DOM-element te schrijven, bevat de tekenreeks mogelijk (JavaScript-) programmacode die onveilige bewerkingen kan uitvoeren. Tijdens het laden van de inhoud is er echter geen risico dat externe tekenreeksen in het DOM-element worden ingevoegd.

Eén beperking betreft het gebruik van de JavaScript-functie `eval()`. Nadat code in de toepassingssandbox is geladen en de gebeurtenishandler onload is verwerkt, kunt u de functie `eval()` slechts met bepaalde beperkingen gebruiken. De volgende regels gelden voor het gebruik van de functie `eval()` *nadat* code is geladen uit de toepassingsbeveiligingssandbox:

- Expressies met literalen zijn toegestaan. Bijvoorbeeld:

```
eval("null");
eval("3 + .14");
eval("'foo'");
```

- Objectliteralen zijn toegestaan, zoals in de volgende situatie:

```
{ prop1: val1, prop2: val2 }
```

- Setter/getters met objectliteralen zijn *niet toegestaan*, zoals in de volgende situatie:

```
{ get prop1() { ... }, set prop1(v) { ... } }
```

- Arrayliteralen zijn toegestaan, zoals in de volgende situatie:

```
[ val1, val2, val3 ]
```

- Expressies met het lezen van eigenschappen zijn *niet toegestaan*, zoals in de volgende situatie:

```
a.b.c
```

- Het oproepen van functies is *niet toegestaan*.
- Functiedefinities zijn *niet toegestaan*.
- Het instellen van eigenschappen is *niet toegestaan*.
- Functieliteralen zijn *niet toegestaan*.

Tijdens het laden van de code, vóór de gebeurtenis `onload`, en tijdens de uitvoering van de gebeurtenishandlerfunctie `onload` gelden deze beperkingen echter niet voor de inhoud van de toepassingsbeveiligingssandbox.

Bij de volgende code bijvoorbeeld treedt een fout op nadat code is geladen:

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

Dynamisch gegenereerde code, zoals bij het oproepen van de functie `eval()`, zou een veiligheidsrisico zijn als deze toegang kreeg tot de toepassingssandbox. Een toepassing kan bijvoorbeeld onbedoeld een tekenreeks uitvoeren die is geladen vanaf een netwerkdomein en schadelijke code bevat. De code kan bijvoorbeeld bestanden op de computer van de gebruiker verwijderen of wijzigen. Of de inhoud van een lokaal bestand doorsturen naar een niet-vertrouwd netwerkdomein.

Dynamische code kan op de volgende manieren worden gegenereerd:

- door het oproepen van de functie `eval()`;
- door eigenschappen van het type `innerHTML` of DOM-functies te gebruiken om scripttags in te voegen die een script buiten de toepassingsmap laden;
- door eigenschappen van het type `innerHTML` of DOM-functies te gebruiken om scripttags in te voegen die inline code bevatten (in plaats van een script te laden via het kenmerk `src`);
- door het kenmerk `src` voor `script` tags zo in te stellen dat een JavaScript-bestand wordt geladen dat zich buiten de toepassingsmap bevindt;
- door het URL-schema `javascript` te gebruiken (zoals in `href="javascript:alert('Test')"`);
- door de functie `setInterval()` of `setTimeout()` te gebruiken, waarbij de eerste parameter (die de functie asynchroon laat uitvoeren) een (te evalueren) tekenreeks en geen functienaam is (zoals in `setTimeout('x = 4', 1000)`);
- door `document.write()` of `document.writeln()` op te roepen.

Code in de toepassingsbeveiligingssandbox kan tijdens het laden van inhoud alleen deze methoden gebruiken.

Deze beperkingen blokkeren het gebruik van *eval()* met JSON-objectliteralen `niet`. Hierdoor kan de inhoud van uw toepassing met de JSON JavaScript-bibliotheek werken. U kunt echter geen overbelaste JSON-code (met gebeurtenishandlers) gebruiken.

Voor andere Ajax-frameworks en JavaScript-codebibliotheken moet u controleren of de code in het framework of de bibliotheek binnen deze beperkingen betreffende dynamisch gegenereerde code werkt. Als dat niet het geval is, neemt u alle inhoud die gebruikmaakt van het framework of de bibliotheek op in een niet-toepassingsbeveiligingssandbox. Zie "Beperkingen voor JavaScript in AIR" op pagina 1111 en "Scripting tussen toepassings- en niet-toepassingsinhoud" op pagina 1161. Adobe houdt een lijst bij van Ajax-frameworks waarvan bekend is dat ze de toepassingsbeveiligingssandbox ondersteunen. Ga hiervoor naar http://www.adobe.com/products/air/develop/ajax/features/.

In tegenstelling tot de inhoud van de toepassingsbeveiligingssandbox kan JavaScript-inhoud van een niet-toepassingsbeveiligingssandbox *wel* de functie `eval()` oproepen om op elk gewenst moment dynamisch gegenereerde code uit te voeren.

Beperkingen betreffende de toegang tot AIR API's (voor niet-toepassingssandboxen)

Adobe AIR 1.0 of hoger

JavaScript-code in een niet-toepassingssandbox heeft geen toegang tot het object `window.runtime`, zodat deze code geen AIR API's kan uitvoeren. Als de inhoud van een niet-toepassingsbeveiligingssandbox de volgende code oproept, treedt een TypeError-fout op:

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

Het fouttype is TypeError (niet-gedefinieerde waarde) omdat de inhoud van de niet-toepassingssandbox het object `window.runtime` niet herkent en het als een niet-gedefinieerde waarde beschouwt.

Als u runtime-functionaliteit toegankelijk wilt maken voor de inhoud van een niet-toepassingssandbox, gebruikt u een scriptbridge. Zie "Scripting tussen toepassings- en niet-toepassingsinhoud" op pagina 1161voor meer informatie.

Beperkingen betreffende het gebruik van XMLHttpRequest-oproepen

Adobe AIR 1.0 of hoger

HTML-inhoud in de beveiligingssandbox van de toepassing kan niet tegelijkertijd XMLHttpRequest-methoden gebruiken voor het laden van buiten de toepassingssandbox terwijl de HTML-inhoud wordt geladen en tijdens de `onLoad`-gebeurtenis.

De HTML-inhoud van niet-toepassingsbeveiligingssandboxen kan standaard geen JavaScript XMLHttpRequest-object gebruiken om gegevens te laden van andere domeinen dan het domein waarvan de aanvraag afkomstig is. Een tag van het type `frame` of `iframe` kan een kenmerk van het type `allowcrosscomainxhr` bevatten. Als u dit kenmerk instelt op een waarde die niet leeg is, kan de inhoud van het frame of iframe het Javascript XMLHttpRequest-object gebruiken om gegevens te laden uit andere domeinen dan het domein van de code waarvan de aanvraag afkomstig is:

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

Zie "Scripting tussen de inhoud van verschillende domeinen" op pagina 1155voor meer informatie.

Beperkingen betreffende het laden van CSS-, frame-, iframe- en img-elementen (voor de inhoud van niet-toepassingssandboxen)

Adobe AIR 1.0 of hoger

De HTML-inhoud van externe (netwerk-) beveiligingssandboxen kan alleen inhoud van het type CSS, `frame`, `iframe` of `img` uit externe sandboxen (netwerk-URL's) laden.

De HTML-inhoud van lokaal-met-bestandssysteem, lokaal-met-netwerk of lokaal-vertrouwde sandboxen kan alleen inhoud van het type CSS, frame, iframe of `img` uit lokale sandboxen (niet van toepassings- of externe sandboxen) laden.

Beperkingen betreffende het oproepen van de JavaScript-methode window.open()

Adobe AIR 1.0 of hoger

Als een venster dat door het oproepen van de JavaScript-methode `window.open()` is gemaakt, de inhoud van een niet-toepassingsbeveiligingssandbox weergeeft, begint de titel van het venster met de titel van het hoofdvenster (vanwaaruit het nieuwe venster is geopend), gevolgd door een dubbele punt. U kunt geen code gebruiken om dat deel van de venstertitel buiten het scherm te verplaatsen.

De inhoud van niet-toepassingsbeveiligingssandboxen kan de JavaScript-methode `window.open()` alleen oproepen als reactie op een gebeurtenis die is gestart door een klik met de muisknop of een druk op een toets van het toetsenbord. Hierdoor wordt voorkomen dat niet-toepassingsinhoud vensters creëert die voor bedrog kunnen worden gebruikt (bijvoorbeeld voor phishingaanvallen). Ook kan de gebeurtenishandler voor de muis- of toetsenbordgebeurtenis de methode `window.open()` niet zo instellen dat deze na een vertraging wordt uitgevoerd (bijvoorbeeld door de functie `setTimeout()` op te roepen).

De inhoud van externe (netwerk-) sandboxen kan de methode `window.open()` alleen gebruiken om de inhoud van externe netwerksandboxen te openen. Deze inhoud kan de methode `window.open()` niet gebruiken om de inhoud van de toepassingssandbox of van lokale sandboxen te openen.

Inhoud in de sandbox Lokaal-met-bestandssysteem, de sandbox Lokaal-met-netwerk en de sandbox Lokaal-vertrouwd (zie "Beveiligingssandboxen" op pagina 1108) kan de methode `window.open()` alleen gebruiken om de inhoud van lokale sandboxen te openen. Deze inhoud kan `window.open()` niet gebruiken om de inhoud van de toepassingssandbox of van externe sandboxen te openen.

Fouten bij het oproepen van beperkte code

Adobe AIR 1.0 of hoger

Als u code oproept die niet in een sandbox kan worden gebruikt vanwege deze beveiligingsbeperkingen, geeft de runtime de foutmelding weer dat er in de runtime van Adobe AIR een beveiligingsfout betreffende JavaScript-code in de toepassingsbeveiligingssandbox is opgetreden.

Zie "JavaScript-beveiligingsfouten voorkomen" op pagina 1041voor meer informatie.

Sandboxbescherming bij het laden van HTML-inhoud uit een tekenreeks

Adobe AIR 1.0 of hoger

Met de methode `loadString()` van de klasse HTMLLoader kunt u HTML-inhoud bij de uitvoering maken. De gegevens die u gebruikt als HTML-inhoud kunnen echter beschadigd zijn als de gegevens worden geladen uit een onbeveiligde internetbron. Om deze reden wordt HTML die is gemaakt met behulp van de methode `loadString()`, niet in de toepassingssandbox geplaatst en heeft deze HTML geen toegang tot API's van AIR. U kunt echter wel de eigenschap `placeLoadStringContentInApplicationSandbox` van een HTMLLoader-object instellen op true, zodat HTML die is gemaakt met de methode `loadString()`, in de toepassingssandbox wordt geplaatst. Zie "HTML-inhoud laden vanuit een tekenreeks" op pagina 1040 voor meer informatie.

Scripting tussen de inhoud van verschillende domeinen

Adobe AIR 1.0 of hoger

Bij de installatie van AIR-toepassingen worden hieraan speciale rechten toegewezen. Het is van groot belang dat deze rechten niet aan andere inhoud worden doorgegeven, zoals externe bestanden en lokale bestanden die geen deel uitmaken van de toepassing.

Informatie over de AIR-sandboxbridge

Adobe AIR 1.0 of hoger

Inhoud van andere domeinen kan normaal gesproken geen scripts in andere domeinen oproepen. Om AIR-toepassingen te beschermen tegen onbedoelde lekkage van informatie of besturingselementen waarvoor speciale toegangsrechten gelden, gelden de volgende beperkingen voor de inhoud van de beveiligingssandbox `application` (inhoud die met de toepassing is geïnstalleerd):

- Code in de toepassingsbeveiligingssandbox mag geen toegang bieden tot andere sandboxen door de methode `Security.allowDomain()` op te roepen. Het oproepen van deze methode vanuit de toepassingsbeveiligingssandbox veroorzaakt een fout.
- Het importeren van niet-toepassingsinhoud in de toepassingssandbox door de eigenschap `LoaderContext.securityDomain` of `LoaderContext.applicationDomain` te activeren, wordt verhinderd.

Er zijn echter situaties waarin de AIR-hoofdtoepassing inhoud van een extern domein nodig heeft voor gecontroleerde toegang tot scripts in de AIR-hoofdtoepassing, of omgekeerd. Hiervoor biedt de runtime het mechanisme *sandboxbridge*, dat als gateway tussen beide sandboxen fungeert. Een sandboxbridge kan expliciete interactie tussen externe en toepassingsbeveiligingssandboxen bieden.

De sandboxbridge maakt twee objecten toegankelijk, waartoe zowel geladen als ladende scripts toegang hebben:

- Het object `parentSandboxBridge` staat ladende inhoud toe om eigenschappen en functies toegankelijk te maken voor scripts in de geladen inhoud.
- Het object `childSandboxBridge` staat geladen inhoud toe om eigenschappen en functies toegankelijk te maken voor scripts in de ladende inhoud.

Objecten die via de sandboxbridge toegankelijk worden gemaakt, worden doorgegeven op basis van waarde en niet op basis van referentie. Alle gegevens zijn geserialiseerd. Dit betekent dat de objecten die door de ene kant van de bridge toegankelijk zijn gemaakt, niet door de andere kant kunnen worden ingesteld, en dat alle toegankelijke objecten geen specifiek type hebben. Bovendien kunt u alleen eenvoudige objecten en functies toegankelijk maken, geen complexe objecten.

Als onderliggende inhoud een object van het bovenliggende object parentSandboxBridge probeert in te stellen, genereert de runtime een SecurityError-fout. Op dezelfde manier genereert de runtime een SecurityError-fout als bovenliggende inhoud een object van het onderliggende object childSandboxBridge probeert in te stellen.

Voorbeeld van sandboxbridge (SWF)

Adobe AIR 1.0 of hoger

Dit voorbeeld is gebaseerd op een AIR-toepassing voor een muziekwinkel. De toepassing wil externe SWF-bestanden toestaan om de prijs van albums toegankelijk te maken maar wil niet dat het externe SWF-bestand aangeeft of de prijs een opruimingsprijs is. Hiervoor biedt de klasse StoreAPI een methode die de prijs opvraagt maar de opruimingsprijs verbergt. Vervolgens wordt een instantie van deze StoreAPI-klasse toegewezen aan de eigenschap `parentSandboxBridge` van het LoaderInfo-object van het Loader-object dat het externe SWF-bestand laadt.

Dit is de code voor de AIR-muziekwinkel:

```
<?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>
```

Het StoreAPI-object roept de hoofdtoepassing op om de normale albumprijs op te vragen maar geeft “Not available” (Niet beschikbaar) weer wanneer de methode `getSaleAlbumPrice()` wordt opgeroepen. De volgende code definieert de klasse 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();
    }
}
```

De volgende code is een voorbeeld van het SWF-bestand PriceQuoter (Prijsofferte), dat de verkoopprijs van de winkel weergeeft maar de opruimingsprijs niet kan weergeven:

```
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());
        }
    }
}
```

Voorbeeld van sandboxbridge (HTML)

Adobe AIR 1.0 of hoger

Bij HTML-inhoud worden de eigenschappen `parentSandboxBridge` en `childSandboxBridge` toegevoegd aan het JavaScript window-object van een subdocument. Zie "Sandboxbridge-interface instellen" op pagina 1059voor een voorbeeld van hoe u bridgefuncties definieert in HTML-inhoud.

API-toegankelijkheid beperken

Adobe AIR 1.0 of hoger

Bij het toegankelijk maken van sandboxbridges is het belangrijk dat u API's toegankelijk maakt die van een hoog niveau zijn en de mate beperken waarin ze kunnen worden misbruikt. Vergeet niet dat de inhoud die uw bridge-implementatie oproept, door een kwaadwillige persoon kan zijn gewijzigd (bijvoorbeeld via een code-injectie). Dit betekent bijvoorbeeld dat een methode `readFile(path:String)` (die de inhoud van een willekeurig bestand leest) die via een bridge toegankelijk wordt gemaakt, kan worden misbruikt. Het is aan te raden een API `readApplicationSetting()` toegankelijk te maken die geen specifiek pad volgt en een specifiek bestand leest. De meer semantische methode beperkt de schade die een toepassing kan aanbrengen als een deel ervan door een kwaadwillige persoon is gewijzigd.

Meer Help-onderwerpen

"Cross-scripting van inhoud in verschillende beveiligingssandboxen" op pagina 1058

"De sandbox van AIR-toepassingen" op pagina 1110

Naar schijf schrijven

Adobe AIR 1.0 of hoger

Toepassingen die in een webbrowser worden uitgevoerd, hebben slechts beperkte interactie met het lokale bestandssysteem van de gebruiker. Webbrowsers passen beveiligingsmaatregelen toe die zorgen dat de computer van de gebruiker niet in gevaar kan worden gebracht door het laden van webinhoud. Bijvoorbeeld: SWF-bestanden die via Flash Player in een browser worden uitgevoerd, kunnen niet rechtstreeks interageren met bestanden die al op de computer van de gebruiker staan. Gedeelde objecten en cookies kunnen naar de computer van de gebruiker worden geschreven voor het bijhouden van gebruikersvoorkeuren en andere gegevens, maar meer niet. Aangezien AIR-toepassingen in het besturingssysteem worden geïnstalleerd, gebruiken ze een ander beveiligingscontract, dat onder andere de mogelijkheid biedt om te lezen van en te schrijven naar het lokale bestandssysteem.

Deze vrijheid gaat gepaard met een grote verantwoordelijkheid voor de ontwikkelaars. Toevallige kwetsbaarheden van toepassingen brengen niet alleen de functionaliteit van de toepassing in gevaar, maar ook de integriteit van de computer van de gebruiker. Daarom wordt ontwikkelaars aangeraden "Aanbevolen beveiligingsprocedures voor ontwikkelaars" op pagina 1162te lezen.

AIR-ontwikkelaars kunnen bestanden in het lokale bestandssysteem via verschillende URL-schemaconventies lezen/schrijven:

URL-schema	Beschrijving
app:/	Een alias voor de toepassingsmap. Bestanden die via dit pad worden benaderd, worden toegewezen aan de toepassingssandbox en beschikken over alle toegangsrechten die door de runtime worden verleend.
app-storage:/	Een alias voor de lokale opslagmap, gestandaardiseerd door de runtime. Bestanden die via dit pad worden benaderd, worden toegewezen aan een niet-toepassingssandbox.
file:///	Een alias die de hoofddirectory van de vaste schijf van de gebruiker vertegenwoordigt. Aan een bestand dat via dit pad wordt benaderd, wordt een toepassingssandbox toegewezen als het bestand in de toepassingsmap bestaat. Als dat niet het geval is, wordt een niet-toepassingssandbox toegewezen.

Opmerking: *AIR-toepassingen kunnen geen inhoud wijzigen via het URL-schema app:. Bovendien is de toepassingsmap mogelijk alleen-lezen, afhankelijk van de beheerdersinstellingen.*

Als er geen beheerdersrechten voor de computer van de gebruiker vereist zijn, mogen AIR-toepassingen naar een willekeurige locatie op de vaste schijf van de gebruiker schrijven. Ontwikkelaars wordt aangeraden het pad `app-storage:/` te gebruiken voor lokale opslag betreffende hun toepassing. Bestanden die vanuit een toepassing naar `app-storage:/` worden geschreven, worden in een standaardlocatie geplaatst:

- In Mac OS: de opslagmap van een toepassing is `<appData>/<appId>/Local Store/`, waarbij `<appData>` de voorkeursmap van de gebruiker is. Dit is doorgaans `/Users/<gebruiker>/Library/Preferences`.
- In Windows: de opslagmap van een toepassing is `<appData>\<appId>\Local Store\`, waarbij `<appData>` de CSIDL_APPDATA 'speciale map' van de gebruiker is. Dit is doorgaans `C:\Documents and Settings\<gebruikersnaam>\Application Data`.
- In Linux: `<appData>/<appID>/Local Store/`, waarbij `<appData>/home/< gebruiker>/.appdata` is

Als een toepassing is ontworpen om te interageren met bestaande bestanden in het bestandssysteem van de gebruiker, moet u "Aanbevolen beveiligingsprocedures voor ontwikkelaars" op pagina 1162lezen.

Veilig werken met niet-vertrouwde inhoud

Adobe AIR 1.0 of hoger

Inhoud die niet aan de toepassingssandbox is toegewezen, kan uw toepassing aanvullende scriptfunctionaliteit bieden, maar alleen als wordt voldaan aan de beveiligingscriteria van de runtime. In dit onderwerp wordt het AIR-beveiligingscontract met niet-toepassingsinhoud besproken.

Security.allowDomain()

Adobe AIR 1.0 of hoger

AIR-toepassingen beperken de toegang van scripts voor niet-toepassingsinhoud in hogere mate dan de Flash Player plug-in voor de browser de toegang van scripts voor niet-vertrouwde inhoud. Bijvoorbeeld: wanneer in Flash Player in de browser een SWF-bestand dat is toegewezen aan de `lokaal-vertrouwde` sandbox de methode `System.allowDomain()` aanroept, wordt scripttoegang verleend voor een willekeurig SWF-bestand dat vanaf het opgegeven domein wordt geladen. Deze benadering is niet toegestaan voor `toepassingsinhoud` in AIR-toepassingen omdat hierdoor onredelijke toegang tot het niet-toepassingsbestand in het bestandssysteem van de gebruiker zou worden verleend. Externe bestanden hebben nooit rechtstreekse toegang tot de toepassingssandbox, ongeacht de aanwezigheid van oproepen van de methode `Security.allowDomain()`.

Scripting tussen toepassings- en niet-toepassingsinhoud

Adobe AIR 1.0 of hoger

Voor AIR-toepassingen die scripts gebruiken tussen toepassings- en niet-toepassingsinhoud, gelden complexere beveiligingsmaatregelen. Bestanden die zich niet in de toepassingssandbox bevinden, hebben alleen via een sandboxbridge toegang tot de eigenschappen en methoden van bestanden in de toepassingssandbox. Een sandboxbridge is een gateway tussen toepassings- en niet-toepassingsinhoud, en maakt expliciete interactie tussen de twee bestanden mogelijk. Wanneer een sandboxbridges correct wordt gebruikt, biedt deze een extra beveiligingslaag door te zorgen dat niet-toepassingsinhoud geen toegang heeft tot objectverwijzingen die deel uitmaken van toepassingsinhoud.

Het voordeel van sandboxbridges wordt het best met een voorbeeld geïllustreerd. Dit voorbeeld is gebaseerd op een AIR-toepassing voor een muziekwinkel. De winkel wil adverteerders die hun eigen SWF-bestanden willen maken, een API bieden waarmee de toepassing van de winkel vervolgens kan communiceren. De winkel wil adverteerders methoden bieden waarmee ze artiesten en cd's in de winkel kunnen opzoeken, maar wil om veiligheidsredenen ook bepaalde methoden en eigenschappen onbereikbaar maken voor de SWF-bestanden van de adverteerders.

Een sandboxbridge kan hiervoor zorgen. Inhoud die extern in een AIR-toepassing wordt geladen tijdens het gebruik van de runtime, heeft standaard geen toegang tot methoden of eigenschappen in de hoofdtoepassing. Met een aangepaste sandboxbridge-implementatie kan een ontwikkelaar services voor de externe inhoud bieden zonder deze methoden of eigenschappen toegankelijk te maken. De sandboxbridge is als een pad tussen vertrouwde en niet-vertrouwde inhoud dat communicatie tussen de ladende en de geladen inhoud mogelijk maakt zonder objectverwijzingen toegankelijk te maken.

Zie “Scripting tussen de inhoud van verschillende domeinen” op pagina 1155voor meer informatie over hoe u sandboxbridges veilig gebruikt.

Bescherming tegen dynamisch gegenereerde, onveilige SWF-inhoud

Adobe AIR 1.0 of hoger

De methode `Loader.loadBytes()` biedt een toepassing de mogelijkheid om SWF-inhoud te genereren op basis van een bytearray. Injectieaanvallen op gegevens die van externe bronnen worden geladen, kunnen echter ernstige schade toebrengen bij het laden van inhoud. Dit geldt met name bij het laden van gegevens in de toepassingssandbox, waar de gegenereerde SWF-inhoud toegang heeft tot de volledige set van AIR API's.

Er zijn echter ook situaties waarin het gebruik van de methode `loadBytes()` zonder het genereren van SWF-programmacode vereist is. U kunt de methode `loadBytes()` gebruiken om grafische gegevens te genereren, bijvoorbeeld om de beeldweergavetiming te besturen. Er zijn ook situaties waarin *wel* programmacode mag worden gegenereerd, zoals bij het dynamisch genereren van SWF-inhoud voor het afspelen van audio. In AIR staat de methode `loadBytes()` u standaard *niet* toe om SWF-inhoud te laden; u mag alleen grafische inhoud laden. In AIR heeft de eigenschap `loaderContext` van de methode `loadBytes()` een eigenschap `allowLoadBytesCodeExecution`, die u op `true` kunt instellen om de toepassing expliciet toe te staan `loadBytes()` te gebruiken om SWF-programmacode te laden. In de volgende code wordt getoond hoe u deze functie gebruikt:

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

Als u `loadBytes()` oproept om SWF-inhoud te laden en de eigenschap `allowLoadBytesCodeExecution` van het LoaderContext-object is ingesteld op `false` (dit is de standaardinstelling), genereert het Loader-object een SecurityError-fout.

Opmerking: *In een toekomstige versie van Adobe AIR wordt deze API mogelijk aangepast. In dat geval moet u mogelijk de inhoud die gebruikmaakt van de eigenschap* `allowLoadBytesCodeExecution` *van de klasse LoaderContext, opnieuw compileren.*

Aanbevolen beveiligingsprocedures voor ontwikkelaars

Adobe AIR 1.0 of hoger

AIR-toepassingen zijn weliswaar gebaseerd op webtechnologieën maar ontwikkelaars mogen niet vergeten dat ze niet in de beveiligingssandbox van de browser werken. Dit betekent dat het mogelijk is om AIR-toepassingen te ontwikkelen die bedoeld of onbedoeld schade kunnen toebrengen aan het systeem. AIR probeert dit risico tot het minimum te beperken maar er zijn altijd wel manieren om kwetsbaarheden te genereren. In dit onderwerp worden belangrijke potentiële kwetsbaarheden besproken.

Risico bij het importeren van bestanden in de toepassingsbeveiligingssandbox

Adobe AIR 1.0 of hoger

Bestanden in de toepassingsmap zijn toegewezen aan de toepassingssandbox en beschikken over alle toegangsrechten van de runtime. Toepassingen die naar het lokale bestandssysteem schrijven, wordt aangeraden naar `app-storage:/` te schrijven. Aangezien deze directory apart van de toepassingsbestanden op de computer van de gebruiker is geplaatst, zijn de bestanden niet toegewezen aan de toepassingssandbox en vormen ze een kleiner veiligheidsrisico. Ontwikkelaars wordt aangeraden het volgende in acht te nemen:

- Neem alleen indien nodig een bestand op in een AIR-bestand (in de geïnstalleerde toepassing).
- Neem alleen een scriptbestand op in een AIR-bestand (in de geïnstalleerde toepassing) als het gedrag ervan volledig begrepen en vertrouwd wordt.
- Schrijf niet naar de toepassingsmap en wijzig de inhoud ervan niet. De runtime genereert een SecurityError-fout om te zorgen dat toepassingen geen bestanden en mappen via het URL-schema `app:/` kunnen wijzigen of ernaar schrijven.
- Gebruik geen gegevens van een netwerkbron als parameters voor AIR API-methoden die tot de uitvoering van code kunnen leiden. Dit geldt onder andere voor het gebruik van de methode `Loader.loadBytes()` en de JavaScript-functie `eval()`.

Risico bij het gebruiken van een externe bron voor het bepalen van paden

Adobe AIR 1.0 of hoger

Een AIR-toepassing kan in gevaar worden gebracht door het gebruik van externe gegevens of inhoud. Daarom moet extra worden opgelet bij het gebruik van gegevens van het netwerk of het bestandssysteem. De verantwoordelijkheid voor het geven van vertrouwen ligt uiteindelijk bij de ontwikkelaars en de netwerkverbindingen die ze maken, maar het laden van externe gegevens is altijd gevaarlijk en kan beter niet worden gedaan voor het invoeren van gegevens bij bewerkingen met vertrouwelijke gegevens. Ontwikkelaars wordt het volgende afgeraden:

- gegevens van een netwerkbron gebruiken om een bestandsnaam te bepalen;
- gegevens van een netwerkbron gebruiken om een URL te definiëren die door de toepassing wordt gebruikt om persoonlijke informatie te verzenden.

Risico bij het gebruiken, opslaan of verzenden van onveilige gebruikersgegevens

Adobe AIR 1.0 of hoger

Het opslaan van gebruikersgegevens in het lokale bestandssysteem van de gebruiker brengt altijd het risico met zich mee dat deze gegevens door een kwaadwillige persoon zijn gewijzigd. Ontwikkelaars wordt aangeraden het volgende in acht te nemen:

- Als gebruikersgegevens lokaal moeten worden opgeslagen, moeten ze worden gecodeerd bij het schrijven naar het lokale bestandssysteem. Via de klasse EncryptedLocalStore biedt de runtime unieke gecodeerde opslag voor elke geïnstalleerde toepassing. Zie "Lokale gegevens gecodeerd opslaan" op pagina 750 voor meer informatie.
- Verzend niet-gecodeerde aanmeldingsgegevens voor een gebruiker alleen naar een netwerkbron als deze betrouwbaar is en als de verzending gebruik maakt van het HTTPS- of TLS-protocol (Transport Layer Security).
- Stel nooit een standaardwachtwoord in bij het creëren van gebruikersgegevens. Laat de gebruiker zijn of haar eigen wachtwoord instellen. Gebruikers die de standaardinstellingen niet wijzigen, stellen hun referenties bloot aan een hacker die het standaardwachtwoord kent.

Risico van een downgradeaanval

Adobe AIR 1.0 of hoger

Tijdens de installatie van een toepassing controleert de runtime of de toepassing momenteel niet is geïnstalleerd. Als de toepassing al is geïnstalleerd, vergelijkt de runtime het versienummer met dat van de geïnstalleerde versie. Als beide versies niet hetzelfde nummer hebben, kan de gebruiker desgewenst de bestaande installatie upgraden. De runtime garandeert echter niet dat de nieuwe versie recenter is dan de geïnstalleerde versie, alleen dat het een andere versie is. Een aanvaller kan de gebruiker een oudere versie sturen om gebruik te maken van deze kwetsbaarheid. Daarom wordt de ontwikkelaar aangeraden een versiecontrole uit te voeren bij het starten van de toepassing. Het is een goed idee om toepassingen het netwerk te laten doorzoeken op eventuele updates. Op die manier wordt onmiddellijk een verouderde versie gedetecteerd, zelfs als een aanvaller erin geslaagd is de gebruiker een oude versie te laten installeren. Ook het gebruik van een duidelijk versieschema voor uw toepassing maakt het moeilijk om gebruikers onbedoeld een oudere versie te laten installeren.

Codeondertekening

Adobe AIR 1.0 of hoger

Alle AIR-installatiebestanden moeten zijn voorzien van codeondertekening. Codeondertekening is een cryptografisch proces waarbij wordt bevestigd dat de opgegeven oorsprong van software correct is. AIR-toepassingen kunnen worden ondertekend door een certificaat dat is uitgegeven door een externe certificeringsinstantie of door een certificaat dat u zelf hebt gemaakt en ondertekend. Een commercieel certificaat van een bekende CI wordt ten zeerste aanbevolen en biedt uw gebruikers de garantie dat ze uw toepassing installeren, en geen vervalste versie. U kunt zelfondertekende certificaten genereren via `adt` uit de SDK of met behulp van Flash, Flash Builder of een andere toepassing die `adt` gebruikt voor het genereren van certificaten. Zelfondertekende certificaten bieden geen enkele garantie dat de geïnstalleerde toepassing echt is en mogen alleen worden gebruikt bij het testen van een toepassing voordat deze op de markt wordt gebracht.

Beveiliging op Android-apparaten

Adobe AIR 2.5 of hoger

AIR past zich op Android net als op alle andere apparaten aan het native beveiligingsmodel aan. Tegelijkertijd houdt AIR zich aan de eigen beveiligingsregels die zijn ontworpen om het ontwikkelaars gemakkelijk te maken om veilige internettoepassingen te schrijven.

Aangezien AIR-toepassingen op Android gebruikmaken van de Android-pakketindeling, valt de installatie onder het Android-beveiligingsmodel. Het installatieprogramma van de AIR-toepassing wordt niet gebruikt.

Het Android-beveiligingsmodel bestaat uit drie primaire aspecten:

- Bevoegdheden
- Handtekeningen van toepassingen
- Gebruiker-id's van toepassingen

Android-machtigingen

Vele Android-functies worden beschermd door het bevoegdhedenmechanisme van het besturingssysteem. Om een beschermde functie te gebruiken, moet de descriptor van de AIR-toepassing declareren dat de toepassing de vereiste bevoegdheid nodig heeft. Wanneer een gebruiker probeert de toepassing te installeren, geeft het Android-besturingssysteem alle gewenste bevoegdheden weer voordat de installatie wordt uitgevoerd.

De meeste AIR-toepassingen moeten Android-machtigingen opgeven in het descriptorbestand van de toepassing. Standaard zijn geen bevoegdheden opgenomen. De volgende bevoegdheden zijn vereist voor beveiligde Android-functies die via de AIR-runtime worden getoond:

ACCESS_COARSE_LOCATION Hiermee geeft u de toepassing via de klasse Geolocation toegang tot locatiegegevens van het WiFi- en mobiele-telefoonnetwerk.

ACCESS_FINE_LOCATION Hiermee krijgt de toepassing via de klasse Geolocation toegang tot GPS-gegevens.

ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE Hiermee geeft u de toepassing via de klasse NetworkInfo toegang tot netwerkgegevens.

CAMERA Hiermee krijgt de toepassing toegang tot de camera.

INTERNET Hiermee kan de toepassing netwerkverzoeken indienen en wordt foutopsporing op afstand mogelijk gemaakt.

READ_PHONE_STATE Hiermee kan de AIR-runtime het geluid dempen in geval van een binnenkomende oproep.

RECORD_AUDIO Hiermee krijgt de toepassing toegang tot de microfoon.

WAKE_LOCK en DISABLE_KEYGUARD Hiermee voorkomt de toepassing via de instellingen voor de klasse SystemIdleMode dat het apparaat inactief wordt.

WRITE_EXTERNAL_STORAGE Hiermee kan de toepassing naar de externe geheugenkaart op het apparaat schrijven.

Handtekeningen van toepassingen

Alle toepassingspakketten die voor het Android-platform worden gemaakt, moeten worden ondertekend. Aangezien AIR-toepassingen voor Android in de native APK-indeling van Android worden opgenomen, worden ze ondertekend in overeenkomst met de Android-regels in plaats van met de AIR-regels. Android en AIR maken vergelijkbaar gebruik van codeondertekeningen, maar er zijn enkele belangrijke verschillen:

- Op Android controleert de ondertekening of de privésleutel in bezit is van de ontwikkelaar. De identiteit van de ontwikkelaar wordt echter niet geverifieerd.

- Het certificaat van Android-toepassingen dient minstens 25 jaar geldig te zijn.
- Android biedt geen ondersteuning voor het migreren van de pakketondertekening naar een ander certificaat. Wanneer een update door een ander certificaat wordt ondertekend, dienen gebruikers de oorspronkelijke toepassing te verwijderen voordat ze de bijgewerkte toepassing kunnen installeren.
- Twee toepassingen die met hetzelfde certificaat zijn ondertekend, kunnen een gedeelde id opgeven die ze toestemming geeft elkaars cache- en gegevensbestanden te openen. (Het delen van deze bestanden is niet mogelijk op AIR.)

Gebruiker-id's van toepassingen

Android maakt gebruik van een Linux-kernel. Aan elke geïnstalleerde toepassing wordt een gebruikers-id van het Linux-type toegewezen die de bevoegdheden voor bewerkingen als het openen van bestanden bepaalt. Bestanden in de toepassingsmap, tijdelijke mappen en de opslagmap zijn beveiligd tegen benadering door bestandssysteembevoegdheden. Naar externe opslag (met andere woorden: naar een SD-kaart) geschreven bestanden kunnen worden gelezen, gewijzigd en verwijderd door andere toepassingen of door de gebruiker wanneer de SD-kaart als een apparaat voor massaopslag wordt aangesloten op een computer.

Met internetverzoeken ontvangen cookies worden niet uitgewisseld door AIR-toepassingen.

Privacy achtergrondafbeelding

Wanneer een gebruiker een toepassing op de achtergrond plaatst, leggen enkele Android-versies een schermafbeelding vast die als miniatuur in de lijst met onlangs gebruikte toepassingen wordt gebruikt. Deze schermafbeelding wordt opgeslagen in het apparaatgeheugen en kan worden geopend door een aanvaller die de fysieke besturing heeft van het apparaat.

Als in uw toepassing gevoelige informatie wordt weergegeven, moet u ervoor zorgen dat dergelijke informatie niet wordt vastgelegd in de schermafbeelding van de achtergrond. De gebeurtenis `deactivate` die door het object NativeApplication wordt verzonden, geeft aan dat een toepassing op het punt staat om naar de achtergrond te worden verplaatst. Gebruik deze gebeurtenis om eventuele gevoelige informatie te wisssen of te verbergen.

Meer Help-onderwerpen

Android: beveiliging en bevoegdheden

Gecodeerde gegevens op Android

AIR-toepassingen op Android kunnen de beschikbare coderingsopties in de geïntegreerde SQL-database gebruiken om gecodeerde gegevens op te slaan. Voor optimale beveiliging baseert u de coderingssleutel op een door de gebruiker ingevoerd wachtwoord dat altijd wordt ingevoerd wanneer de toepassing wordt gestart. Het is lastig of onmogelijk een lokaal opgeslagen coderingssleutel of wachtwoord te verbergen voor een hacker die toegang heeft tot de toepassingsbestanden. Als een hacker de sleutel kan ophalen, biedt het coderen van de gegevens niet meer bescherming dan de op de gebruikers-id gebaseerde bestandssysteembeveiliging van het Android-systeem.

Met de klasse EncryptedLocalStore kunt u gegevens opslaan, maar deze gegevens worden niet gecodeerd op Android-apparaten. Android-beveiliging is in plaats daarvan gebaseerd op de gebruikers-id van de toepassing die de gegevens moet beschermen tegen andere toepassingen. Toepassingen die een gemeenschappelijke gebruikers-id gebruiken en die zijn ondertekend met hetzelfde certificaat, maken gebruik van dezelfde gecodeerde lokale opslag.

Belangrijk: *Op een telefoon met rooting kan elke toepassing met rootingbevoegdheden de bestanden uit alle andere toepassingen openen. Gegevens die aan de hand van de gecodeerde lokale opslag worden opgeslagen, zijn dus niet veilig op een apparaat met rooting.*

Beveiliging op iOS-apparaten

AIR houdt zich op iOS aan het native beveiligingsmodel. Tegelijkertijd houdt AIR zich aan de eigen beveiligingsregels die zijn ontworpen om het ontwikkelaars gemakkelijk te maken om veilige internettoepassingen te schrijven.

Aangezien AIR-toepassingen op iOS gebruikmaken van de iOS-pakketindeling, valt de installatie onder het iOS-beveiligingsmodel. Het installatieprogramma van de AIR-toepassing wordt niet gebruikt. Bovendien wordt geen afzonderlijke AIR-runtime gebruikt op iOS-apparaten. Elke AIR-toepassing beschikt over alle vereiste code om te kunnen functioneren.

Handtekeningen van toepassingen

Alle toepassingspakketten die voor het iOS-platform worden gemaakt, moeten worden ondertekend. Aangezien AIR-toepassingen op iOS in de native iOS IPA-indeling worden opgenomen in pakketten, worden ze ondertekend in overeenkomst met iOS-vereisten in plaats van met AIR-vereisten. iOS en AIR maken vergelijkbaar gebruik van codeondertekeningen, maar er zijn enkele belangrijke verschillen:

- Op iOS moet een door Apple uitgegeven certificaat worden gebruikt voor het tekenen van een toepassing. Door andere certificeringsinstanties uitgegeven certificaten kunnen niet worden gebruikt.
- Op iOS zijn door Apple uitgegeven distributiecertificaten doorgaans één jaar geldig.

Privacy achtergrondafbeelding

Wanneer een gebruiker in iOS een toepassing naar de achtergrond verplaatst, legt het besturingssysteem een schermafbeelding vast die het systeem gebruikt om de overgang te animeren. Deze schermafbeelding wordt opgeslagen in het apparaatgeheugen en kan worden geopend door een aanvaller die de fysieke besturing heeft van het apparaat.

Als in uw toepassing gevoelige informatie wordt weergegeven, moet u ervoor zorgen dat dergelijke informatie niet wordt vastgelegd in de schermafbeelding van de achtergrond. De gebeurtenis `deactivate` die door het object NativeApplication wordt verzonden, geeft aan dat een toepassing op het punt staat om naar de achtergrond te worden verplaatst. Gebruik deze gebeurtenis om eventuele gevoelige informatie te wisssen of te verbergen.

Hoofdstuk 65: ActionScript-voorbeelden gebruiken

Een van de beste manieren om te leren hoe bepaalde klassen en methoden werken, is het uitvoeren van een ActionScript 3.0-codevoorbeeld. U kunt de voorbeelden op verschillende manieren gebruiken, afhankelijk van de apparaten die u gebruikt of waarop u zich richt.

Computers met Flash Professional of Flash Builder Zie "ActionScript 3.0-voorbeelden uitvoeren in Flash Professional" op pagina 1169 of "ActionScript 3.0-voorbeelden uitvoeren in Flash Builder" op pagina 1170 om te lezen hoe u deze ontwikkelomgevingen kunt gebruiken voor het uitvoeren van ActionScript 3.0-voorbeelden. Gebruik trace-instructies en andere tools voor foutopsporing om meer inzicht te krijgen in de manier waarop een codevoorbeeld functioneert.

Mobiele apparatuur U kunt de ActionScript 3.0-codevoorbeelden uitvoeren op mobiele apparaten die ondersteuning bieden voor Flash Player 10.1 en later. Zie "ActionScript 3.0-voorbeelden uitvoeren op mobiele apparaten" op pagina 1172. U kunt deze voorbeelden met gebruik van Flash Professional of Flash Builder ook uitvoeren op uw computer.

TV-apparaten Hoewel u deze voorbeelden niet kunt uitvoeren op een televisie, kunt u er wel iets van opsteken door ze uit te voeren op uw computer. Zie Flash Platform for TV op de Adobe Developer Connection-website voor informatie over het ontwikkelen van toepassingen voor tv's.

Typen voorbeelden

De typen voorbeelden van ActionScript 3.0-code zijn:

- Codefragmentvoorbeelden (deze vindt u verspreid door de volledige documentatie van ActionScript 3.0)
- Op klasse gebaseerde voorbeelden (deze treft u voornamelijk aan in de Naslaggids voor ActionScript 3.0)
- Praktische voorbeelden met meerdere bronbestanden (download gecomprimeerde bronbestanden van www.adobe.com/go/learn_programmingAS3samples_flash_nl)

Codefragmentvoorbeelden

Een codefragmentvoorbeeld ziet er als volgt uit:

```
var x:int = 5;
trace(x); // 5
```

Codefragmenten bevatten code om één enkel idee te demonstreren. De fragmenten bevatten gewoonlijk geen pakket- of klasseninstructies.

Voorbeelden die zijn gebaseerd op een klasse

Veel voorbeelden tonen de broncode van een complete ActionScript-klasse. Een op een klasse gebaseerd voorbeeld ziet er als volgt uit:

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

De code voor een op een klasse gebaseerd voorbeeld bevat een pakketinstructie, een klassendeclaratie en een constructorfunctie.

Praktische voorbeelden met meerdere bronbestanden

Veel van de onderwerpen in de ActionScript 3.0-ontwikkelaarsgids bevatten realistische voorbeelden die laten zien hoe bepaalde ActionScript-functies in een praktische context kunnen worden gebruikt. Deze voorbeelden bevatten gewoonlijk meerdere bestanden, waaronder:

- Een of meerdere ActionScript-bronbestanden
- Een .FLA-bestand dat met Flash Professional kan worden gebruikt
- Een of meerdere MXML-bestanden die met Flash Builder kunnen worden gebruikt
- Gegevensbestanden, afbeeldingsbestanden, geluidsbestanden of andere elementen die door de voorbeeldtoepassing worden gebruikt (optioneel)

Praktische voorbeelden worden gewoonlijk als ZIP-archiefbestanden geleverd.

Lijst met voorbeelden uit de gids voor ontwikkelaars in het ZIP-bestand

Het ZIP-bestand voor Flash Professional CS5 en Flex 4 (te downloaden van www.adobe.com/go/learn_programmingAS3samples_flash_nl) bevat de volgende voorbeelden:

- AlarmClock (“Voorbeeld van gebeurtenisafhandeling: alarmklok” op pagina 150)
- AlgorithmicVisualGenerator (“Voorbeeld van de API voor tekenen: Algorithmic Visual Generator” op pagina 246)
- ASCIIArt (“Voorbeeld van strings: ASCII-afbeeldingen” op pagina 20)
- CapabilitiesExplorer (“Voorbeeld van mogelijkheden: systeemmogelijkheden bepalen” op pagina 923)
- CustomErrors (“Voorbeeld van foutafhandeling: CustomErrors-toepassing” op pagina 74)
- DisplayObjectTransformer (“Voorbeeld van geometrie: een matrixtransformatie toepassen op een weergaveobject” op pagina 231)
- FilterWorkbench (“Voorbeeld van het filteren van weergaveobjecten: Filter Workbench” op pagina 309)
- GlobalStockTicker (“Voorbeeld: een toepassing voor het volgen van aandelenkoersen internationaliseren” op pagina 1010)
- IntrovertIM_HTML (“Voorbeeld van externe API: communicatie tussen ActionScript en JavaScript in een webbrowser” op pagina 897)
- NewsLayout (“TekstField-voorbeeld: tekstopmaak in krantenstijl” op pagina 408)
- PlayList (“Voorbeeld van arrays: PlayList” op pagina 50)
- PodcastPlayer (“Voorbeeld van geluid: Podcast-speler” op pagina 491)
- ProjectionDragger (“Voorbeeld: Perspectiefprojectie” op pagina 377)
- ReorderByZ (“Matrix3D-objecten gebruiken om de weergavevolgorde te wijzigen” op pagina 382)
- RSSViewer (“Voorbeeld van XML in ActionScript: RSS-gegevens van internet laden” op pagina 121)

- RuntimeAssetsExplorer (“Voorbeeld van een filmclip: RuntimeAssetsExplorer” op pagina 350)
- SimpleClock (“Voorbeeld van datum en tijd: eenvoudige analoge klok” op pagina 6)
- SpinningMoon (“Bitmapvoorbeeld: geanimeerde draaiende maan” op pagina 269)
- SpriteArranger (“Voorbeeld van weergaveobjecten: SpriteArranger” op pagina 216)
- TelnetSocket (“Voorbeeld van TCP-socket: een Telnet-client maken” op pagina 844)
- VideoJukebox (“Voorbeeld van video: videojukebox” op pagina 533)
- WikiEditor (“Voorbeeld van een reguliere expressie: een Wiki-parser” op pagina 97)
- WordSearch (“Voorbeeld van muisinvoer: WordSearch” op pagina 609)

Praktische voorbeelden worden eveneens gebruikt in veel van de artikelen in het Flash Developer Center en het Flex Developer Center.

ActionScript 3.0-voorbeelden uitvoeren in Flash Professional

Volg een van de volgende procedures (afhankelijk van het type voorbeeld) om een voorbeeld uit te voeren met Flash Professional.

Een codefragmentvoorbeeld uitvoeren in Flash Professional

Een codefragmentvoorbeeld uitvoeren in Flash Professional:

1. Selecteer Bestand > Nieuw.
2. Selecteer Flash-document in het dialoogvenster Nieuw document en klik op OK.

   Er wordt een nieuw Flash-venster weergegeven.
3. Klik in het deelvenster Tijdlijn op het eerste frame van de eerste laag.
4. Typ of plak in het deelvenster Handelingen het codefragmentvoorbeeld.
5. Selecteer Bestand > Opslaan. Geef het bestand een naam en klik op OK.
6. Als u het voorbeeld wilt testen, selecteert u Besturing > Film testen.

Een op een klasse gebaseerd voorbeeld uitvoeren in Flash Professional

Een op een klasse gebaseerd voorbeeld uitvoeren in Flash Professional:

1. Selecteer Bestand > Nieuw.
2. Selecteer ActionScript-bestand in het dialoogvenster Nieuw document en klik op OK. Er wordt een nieuw editorvenster weergegeven.
3. Kopieer de code voor het op een klasse gebaseerd voorbeeld en plak deze in het editorvenster.

   Als de klasse de hoofddocumentklasse voor het programma is, moet deze de klasse MovieClip uitbreiden:

```
import flash.display.MovieClip;
public class Example1 extends MovieClip{
//...
}
```

Zorg er ook voor dat alle klassen waarnaar in het voorbeeld wordt verwezen, worden gedeclareerd aan de hand van `import`-instructies.

4 Selecteer Bestand > Opslaan. Geef het bestand dezelfde naam als de klasse in het voorbeeld (bijvoorbeeld ContextMenuExample.as).

Opmerking: *Sommige van de op klassen gebaseerde voorbeelden, zoals het flashx.textLayout.container.ContainerController-klassenvoorbeeld, bevatten meerdere niveaus in de pakketdeclaratie (`package flashx.textLayout.container.examples {`). Sla voor deze voorbeelden het bestand op in een submap die overeenkomt met de pakketdeclaratie (flashx/textLayout/container/examples) of verwijder de pakketnaam (zodat ActionScript alleen begint met `package {`). Op deze manier kunt u het bestand vanaf een willekeurige locatie testen.*

5 Selecteer Bestand > Nieuw.

6 Selecteer in het dialoogvenster Nieuw document het type Flash-document (ActionScript 3.0) en klik op OK. Er wordt een nieuw Flash-venster weergegeven.

7 Geef in het deelvenster Eigenschappen in het veld Documentklasse de naam van de voorbeeldklasse op. Deze naam moet identiek zijn aan de naam van het ActionScript-bronbestand dat u zojuist hebt opgeslagen (bijvoorbeeld ContextMenuExample).

8 Selecteer Bestand > Opslaan. Geef het FLA-bestand dezelfde naam als de klasse in het voorbeeld (bijvoorbeeld ContextMenuExample.fla).

9 Als u het voorbeeld wilt testen, selecteert u Besturing > Film testen.

Een praktisch voorbeeld uitvoeren in Flash Professional

Praktische voorbeelden worden gewoonlijk als ZIP-archiefbestanden geleverd. Een praktisch voorbeeld uitvoeren in Flash Professional:

1 Pak het archiefbestand uit in een map naar keuze.

2 Selecteer in Flash Professional achtereenvolgens Bestand > Openen.

3 Blader naar de map waar u het archiefbestand hebt uitgepakt. Selecteer het FLA-bestand in deze map en klik op Openen.

4 Als u het voorbeeld wilt testen, selecteert u Besturing > Film testen.

ActionScript 3.0-voorbeelden uitvoeren in Flash Builder

Volg een van de volgende procedures (afhankelijk van het type voorbeeld) om een voorbeeld uit te voeren met Flash Builder.

Een codefragmentvoorbeeld uitvoeren in Flash Builder

Een codefragmentvoorbeeld uitvoeren in Flash Builder:

1 Maak een nieuw Flex-project (selecteer Bestand > Nieuw > Flex-project) of maak een nieuwe MXML-toepassing in een bestaand Flex-project (selecteer Bestand > Nieuw > MXML-toepassing). Geef het project of de toepassing een beschrijvende naam (zoals ContextMenuExample).

2 Open het gegenereerde MXML-bestand en voeg de tag `<mx:Script>` toe.

3 Plak de inhoud van het codefragmentvoorbeeld tussen de tags `<mx:Script>` en `</mx:Script>`. Sla het MXML-bestand op.

4 Als u het voorbeeld wilt uitvoeren, selecteert u Uitvoeren en vervolgens de menu-uitvoeringsoptie voor het primaire MXML-bestand (zoals Uitvoeren > ContextMenuExample uitvoeren).

Een op een klasse gebaseerd voorbeeld uitvoeren in Flash Builder

Een op een klasse gebaseerd voorbeeld uitvoeren in Flash Builder:

1 Selecteer Bestand > Nieuw > ActionScript-project.

2 Geef in het veld Projectnaam de naam van de primaire klasse op (bijvoorbeeld ContextMenuExample). Gebruik de standaardwaarden voor de overige velden (of wijzig de waarden overeenkomstig uw specifieke omgeving). Klik op Voltooien om het project en het primaire ActionScript-bestand te maken.

3 Wis eventuele gegenereerde inhoud uit het ActionScript-bestand. Plak de voorbeeldcode, waaronder pakket- en importinstructies, in het ActionScript-bestand en sla het bestand op.

Opmerking: *Sommige van de op klassen gebaseerde voorbeelden, zoals het flashx.textLayout.container.ContainerController-klassenvoorbeeld, bevatten meerdere niveaus in de pakketdeclaratie (`package flashx.textLayout.container.examples {`). Sla voor deze voorbeelden het bestand op in een submap die overeenkomt met de pakketdeclaratie (flashx/textLayout/container/examples) of verwijder de pakketnaam (zodat ActionScript alleen begint met `package {`). Op deze manier kunt u het bestand vanaf een willekeurige locatie testen.*

4 Als u het voorbeeld wilt uitvoeren, selecteert u Uitvoeren en vervolgens de menu-uitvoeringsoptie voor de primaire ActionScript-klassennaam (zoals Uitvoeren > ContextMenuExample uitvoeren).

Een praktisch voorbeeld uitvoeren in Flash Builder

Praktische voorbeelden worden gewoonlijk als ZIP-archiefbestanden geleverd. Een praktisch voorbeeld uitvoeren in Flash Builder:

1 Pak het archiefbestand uit in een map naar keuze. Geef de map een beschrijvende naam (bijvoorbeeld ContextMenuExample).

2 Selecteer in Flash Builder achtereenvolgens Bestand > Nieuw Flex-project. Klik in de sectie Projectlocatie op Bladeren en selecteer de map met de voorbeeldbestanden. Geef in het veld Projectnaam de mapnaam op (bijvoorbeeld ContextMenuExample). Gebruik de standaardwaarden voor de overige velden (of wijzig de waarden overeenkomstig uw specifieke omgeving). Klik op Volgende om door te gaan.

3 Klik in deelvenster Uitvoer op Volgende om de standaardwaarde te accepteren.

4 Klik in het deelvenster Bronpaden op de knop Bladeren naast het veld voor het primaire toepassingsbestand. Selecteer het primaire MXML-voorbeeldbestand in de voorbeeldmap. Klik op Voltooien om de projectbestanden te maken.

5 Als u het voorbeeld wilt uitvoeren, selecteert u Uitvoeren en vervolgens de menu-uitvoeringsoptie voor het primaire MXML-bestand (zoals Uitvoeren > ContextMenuExample uitvoeren).

ActionScript 3.0-voorbeelden uitvoeren op mobiele apparaten

ActionScript 3.0-codevoorbeelden kunnen worden uitgevoerd op mobiele apparaten die ondersteuning bieden voor Flash Player 10.1. Codevoorbeelden worden doorgaans echter uitgevoerd omdat men wil weten hoe bepaalde klassen en methoden werken. Als dit het geval is, voert u het voorbeeld uit op een niet-mobiel apparaat zoals een desktopcomputer. Op een desktopcomputer kunt u gebruik maken van trace-instructies en andere hulpmiddelen voor foutopsporing in Flash Professional of Flash Builder, die u kunnen helpen het codevoorbeeld beter te begrijpen.

Als u het voorbeeld op een mobiel apparaat wilt uitvoeren, kopieert u de bestanden naar het apparaat of naar een webserver. Doe het volgende om bestanden naar het apparaat te kopiëren en het voorbeeld in de browser uit te voeren:

1. Maak het SWF-bestand door de instructies in "ActionScript 3.0-voorbeelden uitvoeren in Flash Professional" op pagina 1169 of in "ActionScript 3.0-voorbeelden uitvoeren in Flash Builder" op pagina 1170 op te volgen. In Flash Professional maakt u het SWF-bestand door Controleren > Film testen te selecteren. In Flash Builder maakt u het SWF-bestand door uw Flash Builder-project uit te voeren of samen te stellen of door hierin fouten op te sporen.
2. Kopieer het SWF-bestand naar een map op het mobiele apparaat. Gebruik de met het apparaat meegeleverde software om het bestand te kopiëren.
3. In de adresbalk van de browser op het mobiele apparaat typt u file:// en vervolgens de URL van het SWF-bestand. Typ bijvoorbeeld `file:://applications/myExample.swf`.

Doe het volgende om bestanden naar een webserver te kopiëren en het voorbeeld in de browser van het apparaat uit te voeren:

1. Maak een SWF-bestand en een HTML-bestand. Volg eerst de instructies in "ActionScript 3.0-voorbeelden uitvoeren in Flash Professional" op pagina 1169 of in "ActionScript 3.0-voorbeelden uitvoeren in Flash Builder" op pagina 1170 op. In Flash Professional wordt door het selecteren van Controleren > Film testen alleen het SWF-bestand gemaakt. Om beide bestanden te maken, selecteert u in het dialoogvenster Publicatie-instellingen op het tabblad Indelingen zowel Flash als HTML. Selecteer vervolgens Bestand > Publiceren om zowel het HTML-bestand als het SWF-bestand te maken. In Flash Builder maakt u zowel het SWF-bestand als het HTML-bestand door uw Flash Builder-project uit te voeren of samen te stellen of door hierin fouten op te sporen.
2. Kopieer het SWF-bestand het HTML-bestand naar een map op de webserver.
3. In de adresbalk van de browser op het mobiele apparaat typt u het HTTP-adres van het HTML-bestand. Typ bijvoorbeeld `http://www.myWebServer/examples/myExample.html`.

Denk na over de volgende kwesties, voordat u een voorbeeld op een mobiel apparaat uitvoert.

Grootte van het werkgebied

Als u een voorbeeld uitvoert op een mobiel apparaat is het werkgebied veel kleiner dan bij een niet-mobiel apparaat. Bij veel voorbeelden hoeft het werkgebied niet een bepaalde grootte te hebben. Geef bij het maken van het SWF-bestand een werkgebiedgrootte op die geschikt is voor uw apparaat. Geef bijvoorbeeld 176 x 208 pixels op.

Het doel van de praktische voorbeelden in de ActionScript 3.0-ontwikkelaarsgids is de verschillende ActionScript 3.0-concepten en -klassen te illustreren. De gebruikersinterfaces hiervan zijn dusdanig ontworpen dat ze er mooi uit zien en goed werken op een desktop- of een laptopcomputer. Hoewel de voorbeelden werken op mobiele apparaten, zijn het werkgebied en het ontwerp van de interface niet geschikt voor een klein scherm. Adobe raadt u aan de praktische voorbeelden op een computer uit te voeren om zo meer te leren over ActionScript. Vervolgens kunt u de relevante codefragmenten in uw mobiele toepassingen uitvoeren.

Tekstvelden in plaats van trace-instructies

Wanneer u een voorbeeld uitvoert op een mobiel apparaat, kunt u de uitvoer van de trace-instructies van het voorbeeld niet zien. Als u de uitvoer wilt zien, maakt u een instantie van de klasse TextField. Voeg vervolgens de tekst van de trace-instructies toe aan de eigenschap `text` van het tekstveld.

U kunt de volgende functie gebruiken om een tekstveld voor tracering in te stellen:

```
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;
}
```

Voeg deze functie bijvoorbeeld toe aan de documentklasse als een functie van het type private. Traceer vervolgens in andere methoden van de documentklasse gegevens met soortgelijke code zoals hier staat weergegeven:

```
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("");
```

De methode `appendText()` accepteert slechts één waarde als parameter. Deze waarde moet een tekenreeks zijn (een String-instantie of een letterlijke tekenreeks). Wanneer u de waarde wilt afdrukken van een variabele die geen tekenreeks is, moet u de waarde eerst omzetten in een tekenreeks. De eenvoudigste manier om dit te bereiken is het aanroepen van de methode `toString()` van het object:

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

Tekengrootte

In veel voorbeelden worden tekstvelden gebruikt om te helpen bij het illustreren van een concept. In sommige gevallen wordt de leesbaarheid op een mobiel apparaat verbeterd als de tekengrootte in het tekstveld wordt aangepast. Als in een voorbeeld een tekstveldinstantie genaamd `myTextField` wordt gebruikt, kan de tekengrootte hiervan aan de hand van de volgende code worden aangepast:

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

Gebruikersinvoer vastleggen

Het mobiele besturingssysteem en de mobiele browser leggen een aantal gebeurtenissen aangaande gebruikersinvoer vast, die niet worden ontvangen door de SWF-inhoud. Specifiek gedrag is afhankelijk van het besturingssysteem en de browser, maar kan resulteren in onverwacht gedrag wanneer u de voorbeelden op een mobiel apparaat uitvoert. Zie "Hogere prioriteit KeyboardEvent" op pagina 592 voor meer informatie.

Ook zijn de gebruikersinterfaces van veel voorbeelden ontworpen voor een desktop- of een laptopcomputer. Zo zijn de meeste praktische voorbeelden in de ActionScript 3.0-ontwikkelaarsgids bij uitstek geschikt voor weergave op een desktopcomputer. Daarom wordt niet altijd het hele werkgebied weergegeven op het scherm van het mobiele apparaat. De mogelijkheid om door de inhoud van de browser te pannen is afhankelijk van de browser. Bovendien zijn de voorbeelden niet ontworpen voor het afvangen en afhandelen van schuif- of bladergebeurtenissen. Daarom zijn de gebruikersinterfaces van sommige voorbeelden niet geschikt om op een klein scherm te worden uitgevoerd. Adobe raadt u aan de voorbeelden op een computer uit te voeren om zo meer te leren over ActionScript. Vervolgens kunt u de relevante codefragmenten in uw mobiele toepassingen uitvoeren.

Zie "Weergaveobjecten pannen en schuiven" op pagina 189 voor meer informatie.

Afhandeling van focus

Bij sommige voorbeelden moet u zorgen dat een veld de focus krijgt. Als een veld de focus heeft, kunt u bijvoorbeeld tekst invoeren of een knop selecteren. Gebruik het aanwijsapparaat van het mobiele apparaat, bijvoorbeeld de stylus, of uw vinger om een veld de focus te geven. U kunt hiervoor ook de navigatietoetsen van het mobiele apparaat gebruiken. Gebruik de selectietoets van het mobiele apparaat, net zoals u Enter op een computer gebruikt, voor het selecteren van een knop die de focus heeft. Op sommige apparaten kunt u een knop selecteren door er tweemaal op te tikken.

Zie "Focus beheren" op pagina 587 voor meer informatie.

Afhandeling van muisgebeurtenissen

Veel voorbeelden luisteren of er muisgebeurtenissen plaatsvinden. Op een computer doen muisgebeurtenissen zich bijvoorbeeld voor als een gebruiker met de muisaanwijzer over een weergaveobject gaat of met de muisknop op een weergaveobject klikt. Op mobiele apparaten worden gebeurtenissen die het resultaat zijn van het gebruik van een aanwijsapparaat, zoals een stylus, of het gebruik van een vinger, aanraakgebeurtenissen genoemd. In Flash Player 10.1 worden aanraakgebeurtenissen aan muisgebeurtenissen toegewezen. Deze toewijzing zorgt ervoor dat SWF-inhoud die is ontwikkeld voor een oudere versie van Flash Player 10.1, blijft werken. Voorbeelden werken derhalve wanneer een aanwijsapparaat wordt gebruikt bij het selecteren of slepen van een weergaveobject.

Prestaties

Mobiele apparaten zijn minder krachtig dan desktopapparaten. Sommige voorbeelden die de CPU zwaar belasten, werken mogelijk langzaam op een mobiel apparaat. Zo maakt het voorbeeld in "Voorbeeld van de API voor tekenen: Algorithmic Visual Generator" op pagina 246 bij het invoeren van elk frame uitgebreide berekeningen en tekenbewerkingen. Als u dit voorbeeld op een computer uitvoert, worden er verschillende API's voor tekenen gedemonstreerd. Dit voorbeeld is echter niet geschikt voor sommige mobiele apparaten vanwege de prestatiebeperkingen van deze apparaten.

Op *De prestaties voor het Flash-platform optimaliseren* vindt u meer informatie over de prestaties op mobiele apparatuur.

Tips en trucs

Bij de voorbeelden is geen rekening gehouden met de speciale die het ontwikkelen van toepassingen voor mobiele apparaten met zich meebrengt. Deze speciale vereisten zijn het gevolg van de beperkingen van het geheugen en het vermogen van de CPU van mobiele apparaten. Daar komt nog bij dat een gebruikersinterface voor een klein scherm aan andere eisen moet voldoen dan een gebruikersinterface voor een desktopcomputer. Op *http://www.adobe.com/go/learn_optimizing_fp_nlDe prestaties voor het Flash-platform optimaliseren vindt u meer informatie over het ontwikkelen van toepassingen voor mobiele apparatuur.*

Hoofdstuk 66: SQL-ondersteuning in lokale databases

Adobe AIR bevat een SQL-database-engine met ondersteuning voor lokale SQL-databases met vele standaard SQL-functies, die gebruikmaakt van het open-source SQLite-databasesysteem. De runtime geeft niet aan hoe of waar de databasegegevens worden opgeslagen in het bestandssysteem. Elke database bestaat uit één bestand. Een ontwikkelaar kan de locatie in het bestandssysteem opgeven, waar het databasebestand wordt opgeslagen. Eén AIR-toepassing kan één of meerdere afzonderlijke databases (dat wil zeggen afzonderlijke databasebestanden) openen. Dit document schetst de SQL-syntaxis en ondersteuning van gegevenstypen voor lokale SQL-databases van Adobe AIR. Dit document is niet bedoeld als uitgebreid SQL-naslagwerk. In plaats daarvan beschrijft het specifieke details van het SQL-dialect dat Adobe AIR ondersteunt. De runtime ondersteunt het grootste deel van het standaard SQL-dialect SQL-92. Vanwege de vele verwijzingen, websites, boeken en trainingsmaterialen voor het leren van SQL is dit document niet bedoeld als uitgebreid naslagwerk of zelfstudie voor SQL. In plaats daarvan richt dit document zich met name op de door Apollo AIR ondersteunde SQL-syntaxis en de verschillen tussen SQL-92 en het ondersteunde SQL-dialect.

Conventies voor SQL-instructiedefinities

Binnen instructiedefinities in dit document worden de volgende conventies gebruikt:

- Hoofdletters/kleine letters van tekst
  - UPPER CASE - letterlijke SQL-trefwoorden worden geheel in hoofdletters geschreven.
  - lower case - tijdelijke aanduidingstermen of namen van componenten worden geheel in kleine letters geschreven.
- Lettertekens in definities
  - Met ::= wordt een definitie van een component of instructie aangegeven.
- Groepering en wisseltekens
  - De verticale streep (|) wordt gebruikt tussen alternatieve opties en kan worden gelezen als 'of'.
  - Items tussen vierkante haakjes [] zijn optioneel; de haakjes kunnen één item of een reeks alternatieve items bevatten.
  - Haakjes () om een reeks alternatieven (een reeks items die zijn gescheiden met de verticale streep |), geeft een vereiste groep items aan. Dit is een reeks items die de mogelijke waarden voor één vereist item zijn.
- Kwantoren
  - Een plusteken (+) volgend op een item tussen haakjes geeft aan dat het voorafgaande item 1 of meer keren kan voorkomen.
  - Een sterretje (*) volgend op een item tussen vierkante haakjes geeft aan dat het voorafgaande (tussen vierkante haakjes geplaatste) item 0 of meer keren kan voorkomen.
- Letterlijke tekens
  - Een sterretje (*) dat wordt gebruikt in een kolomnaam of tussen de haakjes die volgen op een functienaam, staat voor het sterretje als letterlijk teken, in plaats van de hoeveelheidsbepaler '0 of meer'.
  - . Een punt (.) geeft de punt als letterlijk teken weer.
  - Een komma (,) geeft de komma als letterlijk teken weer

- Een paar haakjes () om één component of item geeft aan dat de haakjes vereiste haakjes als letterlijke tekens zijn.
- Andere lettertekens staan voor de letterlijke tekens, tenzij anders aangegeven.

Ondersteunde SQL-syntaxis

In dit gedeelte wordt de SQL-syntaxis beschreven die wordt ondersteund door de SQL-database-engine van Adobe AIR. Deze vermeldingen bestaan uit diverse soorten uitleg over verschillende typen instructies en componenten, expressies, ingebouwde functies en operatoren. De volgende onderwerpen komen aan bod:

- Algemene SQL-syntaxis
- Instructies voor het manipuleren van gegevens (SELECT, INSERT, UPDATE en DELETE)
- Instructies voor gegevensdefinitie (CREATE, ALTER, en DROP-instructies voor tabellen, indices, weergaven en triggers)
- Speciale instructies en componenten
- Ingebouwde functies (statistische, scalaire en datum-/tijdnotatiefuncties)
- Operatoren
- Parameters
- Niet-ondersteunde SQL-functies
- Aanvullende SQL-functies

Algemene SQL-syntaxis

Naast de specifieke syntaxis voor diverse instructies en expressies gelden de volgende algemene regels voor de SQL-syntaxis:

Hoofdlettergevoeligheid SQL-instructies, waaronder objectnamen, zijn niet hoofdlettergevoelig. Toch worden SQL-instructies vaak geschreven met SQL-trefwoorden in hoofdletters. Deze conventie wordt ook aangehouden in dit document. Hoewel de SQL-syntaxis niet hoofdlettergevoelig is, zijn letterlijke tekstwaarden (zogenaamde literals) wel hoofdlettergevoelig en kunnen vergelijkings- en sorteerbewerkingen ook hoofdlettergevoelig zijn. Dit wordt bepaald door de sorteervolgorde die is gedefinieerd voor een kolom of bewerking. Zie COLLATE voor meer informatie.

Witruimte Een teken voor witruimte (zoals spatie, tab, nieuwe regel, enzovoort) moet worden gebruikt voor het scheiden van afzonderlijke woorden in een SQL-instructie. Witruimte is echter optioneel tussen woorden en symbolen. Het type en de hoeveelheid tekens voor witruimte in een SQL-instructie is niet relevant. U kunt witruimte, zoals inspringen en regeleinden, gebruiken om uw SQL-instructies beter leesbaar te maken, zonder dat dit invloed heeft op de betekenis van de instructie.

Instructies voor het manipuleren van gegevens

Instructies voor het manipuleren van gegevens zijn de meest gebruikte SQL-instructies. Deze instructies worden gebruikt voor het ophalen, toevoegen, bewerken en verwijderen van gegevens uit databasetabellen. De volgende instructies voor het manipuleren van gegevens worden ondersteund: SELECT, INSERT, UPDATE en DELETE.

SELECT

De instructie SELECT wordt gebruikt om een query op de database uit te voeren. Het resultaat van een SELECT-instructie is nul of meer rijen met gegevens, waarbij elke rij een vast aantal kolommen heeft. Het aantal kolommen in het resultaat wordt bepaald door de kolomnaam of expressielijst result tussen SELECT en optionele FROM-trefwoorden.

```
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
```

Elke expressie kan als een resultaat worden gebruikt. Als een resultaatexpressie bestaat uit *, komen alle kolommen van alle tabellen in de plaats van die ene expressie. Als de expressie bestaat uit de naam van een tabel gevolgd door .*, bestaat het resultaat uit alle kolommen in die ene tabel.

Het trefwoord DISTINCT leidt ertoe dat een subset van resultaatrijen wordt geretourneerd, waarin elke resultaatrij anders is. NULL-waarden worden niet als onderling verschillend behandeld. Het standaardgedrag is dat alle resultaatrijen worden geretourneerd, hetgeen expliciet kan worden gemaakt via het trefwoord ALL.

De query wordt uitgevoerd op een of meer tabellen die worden opgegeven na het trefwoord FROM. Als meerdere tabelnamen worden gescheiden met komma's, gebruikt de query de kruislingse koppeling van de diverse tabellen. De JOIN-syntaxis kan ook worden gebruikt om aan te geven hoe tabellen worden gekoppeld. Het enige type outer join dat wordt ondersteund, is LEFT OUTER JOIN. De componentexpressie ON in join-args moet een booleaanse waarde opleveren. Een subquery tussen haakjes kan als een tabel worden gebruikt in de FROM-component. De gehele FROM-component kan worden weggelaten, waardoor in dit geval het resultaat bestaat uit één enkele rij die bestaat uit de waarden van de expressielijst result.

De component WHERE wordt gebruikt on het aantal rijen dat de query ophaalt te beperken. WHERE-componentexpressies moeten een booleaanse waarde opleveren. Filteren met de component WHERE moet vóór groeperen worden uitgevoerd, opdat WHERE-componentexpressies geen statistische functies omvatten.

De GROUP BY-component leidt ertoe dat een of meer rijen uit het resultaat moeten worden gecombineerd in één rij uitvoer. Een GROUP BY-component is met name handig wanneer het resultaat statistische functies bevat. De expressies in de GROUP BY-component hoeven niet per se voor te komen in de expressielijst SELECT.

De component HAVING lijkt in die zin op WHERE, dat deze het door de instructie geretourneerde aantal rijen beperkt. De component HAVING wordt echter toegepast nadat de groepering die wordt opgegeven door een GROUP BY-component heeft plaatsgevonden. Daarom kan de HAVING-expressie verwijzen naar waarden met daarin statistische functies. Een HAVING-componentexpressie hoeft niet per se voor te komen in de lijst SELECT. Evenals bij een WHERE-expressie, moet de HAVING-expressie een booleaanse waarde opleveren.

Door de ORDER BY-component worden de rijen van de uitvoer gesorteerd. Het argument sort-expr-list bij de ORDER BY-component bestaat uit een lijst met expressies die als de sleutel voor de sorteerbewerking worden gebruikt. De expressies hoeven geen onderdeel te zijn van het resultaat van een enkelvoudige SELECT, maar bij een samengestelde SELECT (een SELECT die een van de operatoren compound-op gebruikt) moet elke sorteerexpressie exact overeenkomen met een van de resultaatkolommen. Elke sorteerexpressie kan optioneel worden gevolgd door een sort-order-component die bestaat uit het trefwoord COLLATE en de naam van een sorteerfunctie die wordt gebruikt voor het rangschikken van tekst, en/of het trefwoord ASC of DESC om de sorteervolgorde (oplopend of aflopend) aan te geven. De sort-order kan worden weggelaten, zodat de standaardinstelling (oplopende volgorde) wordt gebruikt. Zie COLLATE voor een definitie van de component COLLATE en de sorteerfuncties.

De component LIMIT definieert een bovengrens voor het aantal rijen dat wordt geretourneerd in het resultaat. Een negatieve LIMIT geeft aan dat er geen bovengrens is. De optionele OFFSET volgend op LIMIT geeft aan hoeveel rijen moeten worden overgeslagen aan het begin van de resultatenset. Bij een samengestelde SELECT-query mag de LIMIT-component alleen voorkomen na de laatste SELECT-instructie, waarbij de limiet geldt voor de gehele query. Denk eraan dat als het trefwoord OFFSET wordt gebruikt in de LIMIT-component, de limiet het eerste gehele getal is en dat de verschuiving (offset) het tweede gehele getal is. Als in plaats van het trefwoord OFFSET een komma wordt gebruikt, is de verschuiving het eerste getal en is de limiet het tweede getal. Deze schijnbare contradictie is doelbewust: hierdoor wordt een maximale compatibiliteit met verouderde SQL-databasesystemen verkregen.

Een samengestelde SELECT wordt gevormd door twee of meer enkelvoudige SELECT-instructies die zijn verbonden door een van de operatoren UNION, UNION ALL, INTERSECT of EXCEPT. In een samengestelde SELECT moeten alle deel uitmakende SELECT-instructies hetzelfde aantal resultaatkolommen opgeven. Er kan slechts een enkelvoudige ORDER BY-component voorkomen na de laatste SELECT-instructie (en vóór de enkelvoudige LIMIT-component als er één is opgegeven). De operatoren UNION en UNION ALL combineren de resultaten van de voorafgaande en de volgende SELECT-instructies in één tabel. Het verschil is dat bij UNION alle resultaatrijen van elkaar verschillen, terwijl er bij UNION ALL duplicaatrijen kunnen voorkomen. De operator INTERSECT neemt het snijpunt van de resultaten van de voorafgaande en volgende SELECT-instructies. EXCEPT neemt het resultaat van de voorafgaande SELECT nadat eerst de resultaten van de volgende SELECT zijn verwijderd. Wanneer er drie of meer SELECT-instructies worden verbonden in een samenstelling, is de volgorde van groepering eerste naar laatste.

Zie Expressies voor een definitie van toegestane expressies.

Vanaf versie AIR 2.5 wordt de SQL CAST-operator ondersteund voor het lezen van BLOB-gegevens die moeten worden omgezet in ActionScript ByteArray-objecten. De volgende code leest bijvoorbeeld onbewerkte gegevens die niet zijn opgeslagen in de AMF-indeling en slaat deze op in een ByteArray-object:

```
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

De instructie INSERT komt voor in twee basisvormen en wordt gebruikt om tabellen te vullen met gegevens.

```
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
```

De eerste vorm (met het trefwoord VALUES) maakt één nieuwe rij in een bestaande tabel. Als er geen column-list wordt opgegeven, moet het aantal waarden gelijk zijn aan het aantal kolommen in de tabel. Als er wel een column-list wordt opgegeven, moet het aantal waarden overeenkomen met het aantal opgegeven kolommen. Kolommen van de tabel die niet voorkomen in de kolommenlijst, worden gevuld met de standaardwaarde die is opgegeven bij het maken van de tabel, of met NULL als er geen standaardwaarde is gedefinieerd.

De tweede vorm van de INSERT-instructie maakt gebruik van gegevens uit een SELECT-instructie. Het aantal kolommen in het resultaat van de SELECT moet exact overeenkomen met het aantal kolommen in de tabel als column-list niet is opgegeven, of moet overeenkomen met het aantal kolommen dat wordt genoemd in de column-list. Er wordt een nieuw item in de tabel gemaakt voor elke rij van het SELECT-resultaat. De SELECT kan enkelvoudig of samengesteld zijn. Zie SELECT voor een definitie van toegestane SELECT-instructies.

Met de optionele conflict-algorithm kan een alternatief algoritme voor het oplossen van beperkingsconflicten worden opgegeven, dat tijdens het gebruik van deze ene instructie moet worden gebruikt. Zie "Speciale instructies en componenten" op pagina 1187 voor een uitleg en definitie van conflictalgoritmen.

De twee vormen van REPLACE INTO van de instructie zijn gelijk aan het gebruik van de standaardvorm van INSERT [OR conflict-algorithm] met het conflictalgoritme REPLACE (d.w.z. de vorm INSERT OR REPLACE...).

De twee vormen van REPLACE INTO van de instructie zijn gelijk aan het gebruik van de standaardvorm van INSERT [OR conflict-algorithm] met het conflictalgoritme REPLACE (d.w.z. de vorm INSERT OR REPLACE...).

UPDATE

De update-opdracht wijzigt de bestaande records in een tabel.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,...
[WHERE expr]
```

De opdracht bestaat uit het trefwoord UPDATE gevolgd door de naam van de tabel waarin u de records wilt bijwerken. Na het trefwoord SET geeft u in een lijst met komma's als scheidingsteken de naam van de kolom op en de waarde waarin de kolom moet worden gewijzigd. De WHERE-componentexpressie verschaft de rij of rijen waarin de records worden bijgewerkt.

DELETE

De instructie DELETE wordt gebruikt om records uit een tabel te verwijderen.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

De instructie bestaat uit de DELETE FROM-trefwoorden gevolgd door de naam van de tabel waaruit records moeten worden verwijderd.

Zonder een WHERE-component worden alle rijen van de tabel verwijderd. Als een WHERE-component wordt opgegeven, worden alleen die rijen verwijderd, die overeenkomen met de expressie. De expressie van de WHERE-component moet een booleaanse waarde opleveren. Zie Expressies voor een definitie van toegestane expressies.

Instructies voor gegevensdefinitie

Instructies voor gegevensdefinitie worden gebruikt om databaseobjecten, zoals tabellen, weergaven, indices en triggers, te maken, bewerken en verwijderen. De volgende instructies voor gegevensdefinitie worden ondersteund:

- Tabellen:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE

- Indices:
  - CREATE INDEX
  - DROP INDEX
- Weergaven:
  - CREATE VIEWS
  - DROP VIEWS
- Triggers:
  - CREATE TRIGGERS
  - DROP TRIGGERS

CREATE TABLE

Een CREATE TABLE-instructie bestaat uit de CREATE TABLE-trefwoorden gevolgd door de naam van de nieuwe tabel, met daarna (tussen haakjes) een lijst met kolomdefinities en -beperkingen. De tabelnaam kan bestaan uit hetzij een ID, hetzij een tekenreeks.

```
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]*
```

Elke kolomdefinitie is de naam van de kolom gevolgd door het gegevenstype voor die kolom, met daarna een of meer optionele kolombeperkingen. Het gegevenstype voor de kolom bepaalt welke gegevens in die kolom kunnen worden opgeslagen. Als wordt gepoogd om een waarde op te slaan in een kolom met een ander gegevenstype, converteert de runtime de waarde indien mogelijk naar het juiste type. Als dit niet mogelijk is, wordt er een fout gesignaleerd. Zie het gedeelte Ondersteuning van gegevenstypen voor aanvullende informatie.

De kolombeperking NOT NULL geeft aan dat de kolom geen NULL-waarden kan bevatten.

Een UNIQUE-beperking leidt ertoe dat er een index voor de opgegeven kolom of kolommen wordt gemaakt. Deze index moet unieke sleutels bevatten: in geen enkele twee rijen mogen dubbele waarden of combinaties van waarden voor de opgegeven kolom of kolommen voorkomen. Een CREATE TABLE-instructie kan meerdere UNIQUE-beperkingen hebben, waaronder meerdere kolommen met een UNIQUE-beperking in de definitie van de kolom en/of meerdere UNIQUE-beperkingen op tabelniveau.

Een CHECK-beperking definieert een expressie die wordt geëvalueerd en die waar moet zijn voordat de gegevens van de rij kunnen worden ingevoegd of bijgewerkt. De CHECK-expressie moet een booleaanse waarde opleveren.

Een COLLATE-component in een kolomdefinitie geeft aan welke tekstsorteerfunctie moet worden gebruikt bij het vergelijken van tekstitems voor de kolom. Standaard wordt de sorteerfunctie BINARY gebruikt. Zie COLLATE voor meer informatie over de component COLLATE en sorteerfuncties.

De DEFAULT-beperking geeft een standaardwaarde aan die moet worden gebruikt bij het uitvoeren van een INSERT. De waarde kan NULL, een tekenreeksconstante of een getal zijn. De standaardwaarde kan ook een van de speciale, niet-hoofdlettergevoelige trefwoorden CURRENT_TIME, CURRENT_DATE of CURRENT_TIMESTAMP zijn. Als de waarde NULL, een tekenreeksconstante of een getal is, wordt deze letterlijk ingevoegd in de kolom telkens wanneer een INSERT-instructie geen waarde voor de kolom opgeeft. Als de waarde CURRENT_TIME, CURRENT_DATE of CURRENT_TIMESTAMP is, wordt de huidige UTC-datum en/of -tijd ingevoegd in de kolom. Voor CURRENT_TIME is de notatie HH:MM:SS. Voor CURRENT_DATE is de notatie YYYY-MM-DD. De notatie voor CURRENT_TIMESTAMP is YYYY-MM-DD HH:MM:SS.

Door op de normale wijze een PRIMARY KEY op te geven wordt alleen een UNIQUE-index voor de daarmee overeenkomende kolom of kolommen gemaakt. Als de PRIMARY KEY-beperking echter geldt voor een enkelvoudige kolom met daarin gegevens van het type INTEGER (of een van de synoniemen zoals int), wordt die kolom intern gebruikt als de daadwerkelijke primaire sleutel voor de tabel. Dit betekent dat de kolom alleen waarden in de vorm van unieke gehele getallen mag bevatten. (In vele SQLite-implementaties leidt alleen het kolomtype INTEGER ertoe dat de kolom als interne primaire sleutel fungeert, maar in Adobe AIR leiden synoniemen voor INTEGER, zoals int, ook tot dat gedrag.)

Als een tabel geen kolom INTEGER PRIMARY KEY heeft, wordt automatisch een sleutel van gehele getallen gegenereerd wanneer een rij wordt ingevoegd. De primaire sleutel voor een rij is altijd toegankelijk met behulp van een van de speciale namen ROWID, OID of _ROWID_. Deze namen kunnen worden gebruikt ongeacht of het een expliciet gedeclareerde INTEGER PRIMARY KEY of een intern gegenereerde waarde betreft. Als de tabel echter een expliciete INTEGER PRIMARY KEY heeft, wordt de werkelijke kolomnaam (en niet de speciale naam) weergegeven in de resultaatgegevens.

Een kolom INTEGER PRIMARY KEY kan ook het trefwoord AUTOINCREMENT bevatten. Wanneer het trefwoord AUTOINCREMENT wordt gebruikt, genereert de database automatisch een sequentieel incrementele sleutel van gehele getallen en voegt deze automatisch in de kolom INTEGER PRIMARY KEY in wanneer een INSERT-instructie wordt uitgevoerd waarbij geen expliciete kolomwaarde wordt opgegeven.

Er mag slechts één PRIMARY KEY-beperking voorkomen in een CREATE TABLE-instructie. Deze kan onderdeel zijn van een definitie van één kolom of uit één enkelvoudige PRIMARY KEY-beperking op tabelniveau bestaan. Een primaire-sleutelkolom is impliciet NOT NULL.

Met de optionele conflict-clause die kan volgen op vele beperkingen, kan een alternatief algoritme voor het oplossen van beperkingsconflicten voor die beperking worden opgegeven. Standaard is dit ABORT. Verschillende beperkingen binnen dezelfde tabel kunnen verschillende algoritmen voor het oplossen van conflicten hebben. Als een INSERT- of UPDATE-instructie een ander algoritme voor het oplossen van conflicten opgeeft, wordt dat algoritme gebruikt, in plaats van het algoritme dat is opgegeven in de MAKE TABLE-instructie. Zie het gedeelte ON CONFLICT in "Speciale instructies en componenten" op pagina 1187 voor meer informatie.

Aanvullende beperkingen, zoals FOREIGN KEY-beperkingen, leiden niet tot een fout, maar worden wel door de runtime genegeerd.

Als het trefwoord TEMP of TEMPORARY voorkomt tussen CREATE en TABLE, is de tabel die wordt gemaakt, alleen zichtbaar binnen dezelfde databaseverbinding (SQLConnection-instantie). Deze wordt automatisch verwijderd wanneer de databaseverbinding wordt gesloten. Alle indices die worden gemaakt voor een tijdelijke tabel, zijn ook tijdelijk. Tijdelijke tabellen en indices worden opgeslagen in een afzonderlijk bestand dat los staat van het hoofddatabasebestand.

Als het optionele prefix (voorvoegsel) database-name wordt opgegeven, wordt de tabel gemaakt in een benoemde database (een database die is verbonden met de SQLConnection-instantie door het aanroepen van de methode attach() met de opgegeven databasenaam). Het is onjuist om zowel een prefix database-name als het trefwoord TEMP op te geven, tenzij het prefix database-name temp is. Als er geen databasenaam wordt opgegeven en als het trefwoord TEMP niet aanwezig is, wordt de tabel gemaakt in de hoofddatabase (de database die was verbonden met de SQLConnection-instantie die de methode open() of openAsync() gebruikt).

Er zijn geen willekeurige limieten voor het aantal kolommen of het aantal beperkingen in een tabel. Ook is er geen willekeurige limiet voor de hoeveelheid gegevens in een rij.

Het formulier CREATE TABLE AS definieert de tabel als de resultatenset van een query. De namen van de tabelkolommen zijn de namen van de kolommen in het resultaat.

Als de optionele component IF NOT EXISTS aanwezig is en er al een andere tabel met dezelfde naam bestaat al, negeert de database de CREATE TABLE-instructie.

Een tabel kan worden verwijderd met de instructie DROP TABLE, terwijl beperkte wijzigingen kunnen worden aangebracht met behulp van de instructie ALTER TABLE.

ALTER TABLE

Met de instructie ALTER TABLE kan de gebruiker de naam van een kolom wijzigen of een nieuwe kolom toevoegen aan een bestaande tabel. Het is niet mogelijk om een kolom te verwijderen uit een tabel.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

De syntaxis RENAME TO wordt gebruikt om de naam te wijzigen van de tabel die wordt aangeduid met [database-name.] table-name in new-table-name. Deze instructie kan niet worden gebruikt om een tabel te verplaatsen tussen gekoppelde databases, maar alleen om de naam te wijzigen van een tabel binnen dezelfde database.

Als de tabel waarvan de naam wordt gewijzigd, triggers of indices heeft, blijven deze gekoppeld aan de tabel, ook nadat de naam ervan is gewijzigd. Als er echter weergavedefinities of instructies worden uitgevoerd door triggers die verwijzen naar de tabel waarvan de naam wordt gewijzigd, worden deze niet automatisch gewijzigd met de nieuwe tabelnaam. Als een tabel met een gewijzigde naam gekoppelde weergaven of triggers heeft, moet u de triggers of weergavedefinities handmatig verwijderen en opnieuw maken met de nieuwe tabelnaam.

De syntaxis ADD [COLUMN] wordt gebruikt om een nieuwe kolom toe te voegen aan een bestaande tabel. De nieuwe kolom wordt altijd toegevoegd aan het einde van de lijst met bestaande kolommen. De component column-def kan elk van de in de instructie CREATE TABLE toegestane vormen hebben, maar hierbij gelden de volgende beperkingen:

- De kolom mag geen beperking van het type PRIMARY KEY of UNIQUE hebben.
- De kolom mag geen standaardwaarde voor CURRENT_TIME, CURRENT_DATE of CURRENT_TIMESTAMP hebben.
- Als een beperking van het type NOT NULL is opgegeven, moet de kolom een andere standaardwaarde dan NULL hebben.

De uitvoeringstijd van de instructie ALTER TABLE wordt niet beďnvloed door de hoeveelheid gegevens in de tabel.

DROP TABLE

De instructie DROP TABLE verwijdert een tabel die is toegevoegd met de instructie CREATE TABLE. De tabel met de opgegeven table-name is de tabel die wordt uitgenomen. Deze wordt volledig verwijderd uit de database en het schijfbestand. De tabel kan niet worden hersteld. Alle indices die aan de tabel zijn gekoppeld, worden ook verwijderd.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

De instructie DROP TABLE verandert de grootte van het databasebestand niet. Witruimte in de database blijft behouden en wordt gebruikt in volgende INSERT-bewerkingen. Gebruik voor het verwijderen van vrije ruimte in de database de methode SQLConnection.clean(). Als de parameter autoClean wordt ingesteld op true wanneer de database voor de eerste keer wordt gemaakt, wordt de ruimte automatisch vrijgemaakt.

De optionele component IF EXISTS onderdrukt de fout die normaliter zou worden gegeven als de tabel niet bestaat.

CREATE INDEX

De instructie CREATE INDEX bestaat uit de trefwoorden CREATE INDEX, gevolgd door de naam van de nieuwe index, het trefwoord ON, de naam van een eerder gemaakte tabel die moet worden geďndexeerd, en een tussen haakjes geplaatste lijst met namen van kolommen in de tabel, waarvan de waarden worden gebruikt voor de indexsleutel.

```
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]
```

Elke kolomnaam kan worden gevolgd door het trefwoord ASC of DESC om de sorteervolgorde aan te geven, maar de sorteervolgordeaanduiding wordt genegeerd door de runtime. Sorteren geschiedt altijd in oplopende volgorde.

De component COLLATE die volgt op elke kolomnaam, definieert een sorteervolgorde die wordt gebruikt voor tekstwaarden in die kolom. De standaardsorteervolgorde is de sorteervolgorde die is gedefinieerd voor die kolom in de instructie CREATE TABLE. Als er geen sorteervolgorde is opgegeven, wordt de sorteervolgorde BINARY gebruikt. Zie COLLATE voor een definitie van de component COLLATE en de sorteerfuncties.

Er zijn geen limieten voor het aantal indices dat aan één tabel kan worden gekoppeld. Ook zijn er geen limieten aan het aantal kolommen in een index.

DROP INDEX

De instructie DROP INDEX verwijdert een index die is toegevoegd met de instructie CREATE INDEX. De opgegeven index wordt volledig verwijderd uit het databasebestand. De enige manier om de index te herstellen is het opnieuw opgeven van de juiste instructie CREATE INDEX.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

De instructie DROP INDEX verandert de grootte van het databasebestand niet. Witruimte in de database blijft behouden en wordt gebruikt in volgende INSERT-bewerkingen. Gebruik voor het verwijderen van vrije ruimte in de database de methode SQLConnection.clean(). Als de parameter autoClean wordt ingesteld op true wanneer de database voor de eerste keer wordt gemaakt, wordt de ruimte automatisch vrijgemaakt.

CREATE VIEW

De instructie CREATE VIEW wijst een naam toe aan een vooraf gedefinieerde instructie SELECT. Deze nieuwe naam kan vervolgens worden gebruikt in een FROM-component van een andere SELECT-instructie in plaats van een tabelnaam. Weergaven worden doorgaans gebruikt voor het vereenvoudigen van query's door het combineren van een samengestelde (en veelgebruikte) set gegevens tot een structuur die kan worden gebruikt in andere bewerkingen.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS
select-statement
```

Als het trefwoord TEMP of TEMPORARY voorkomt tussen CREATE en VIEW, is de weergave die wordt gemaakt alleen zichtbaar voor de SQLConnection-istantie die de database heeft geopend, en wordt de weergave automatisch verwijderd wanneer de database wordt gesloten.

Als een [database-name] wordt opgegeven, wordt de weergave gemaakt in de benoemde database (een database die was verbonden met de SQLConnection-instantie met de methode attach(), met het opgegeven argument name. Het is onjuist om zowel een [database-name] als het trefwoord TEMP op te geven, tenzij de [database-name] gelijk is aan temp. Als er geen databasenaam wordt opgegeven en als het trefwoord TEMP niet aanwezig is, wordt de weergave gemaakt in de hoofddatabase (de database die was verbonden met de SQLConnection-instantie met de methode open() of openAsync()).

Weergaven zijn alleen-lezen. Een instructie DELETE, INSERT of UPDATE kan alleen worden gebruikt voor een weergave als er ten minste één trigger van het bijbehorende type (INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE) is gedefinieerd. Zie CREATE TRIGGER voor informatie over het maken van een trigger voor een weergave.

Een weergave wordt verwijderd uit een database met de instructie DROP VIEW.

DROP VIEW

De instructie DROP VIEW verwijdert een weergave die is gemaakt met de instructie CREATE VIEW.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

De opgegeven view-name is de naam van de weergave die moet worden uitgenomen. Deze wordt verwijderd uit de database, maar in de onderliggende tabellen worden geen gegevens gewijzigd.

CREATE TRIGGER

De instructie CREATE TRIGGER wordt gebruikt om triggers toe te voegen aan het databaseschema. Een trigger is een databasebewerking (de trigger-action) die automatisch wordt uitgevoerd wanneer een opgegeven databasegebeurtenis (de database-event) plaatsvindt.

```
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]*
```

Een trigger moet in werking treden wanneer een DELETE, INSERT of UPDATE van een bepaalde databasetabel plaatsvindt, of telkens wanneer een UPDATE van een of meer opgegeven kolommen van een tabel worden bijgewerkt. Triggers zijn permanent, behalve wanneer het trefwoord TEMP of TEMPORARY is gebruikt. In dat geval wordt de trigger verwijderd wanneer de verbinding met de hoofddatabase van de SQLConnection-instantie wordt gesloten. Als er geen tijdsbepaling is opgegeven (BEFORE of AFTER), wordt de trigger standaard ingesteld op BEFORE.

Alleen triggers van het type FOR EACH ROW worden ondersteund, waardoor de tekst FOR EACH ROW optioneel is. Bij een trigger van het type FOR EACH ROW worden de trigger-stepinstructies uitgevoerd voor elke databaserij die wordt ingevoegd, bijgewerkt of verwijderd door de instructie die de trigger activeert, als de expressie van de WHEN-component wordt geëvalueerd als true.

Als een WHEN-component wordt opgegeven, worden de als triggerstappen opgegeven SQL-instructies alleen uitgevoerd voor rijen waarvoor de WHEN-component waar is. Als er geen WHEN-component is opgegeven, worden de SQL-instructies uitgevoerd voor alle rijen.

Binnen de tekst van een trigger (de component trigger-action) zijn de waarden vóór de wijziging en die van na de wijziging van de onderworpen tabel beschikbaar met de speciale tabelnamen OLD en NEW. De structuur van de tabellen OLD en NEW komt overeen met de structuur van de tabel waarvoor de trigger werd gemaakt. De tabel OLD bevat die rijen die zijn gewijzigd of verwijderd door de activerende instructie, zoals ze waren vóór de bewerkingen van de activerende instructie. De tabel NEW bevat die rijen die zijn gewijzigd of gemaakt door de activerende instructie, zoals ze zijn na de bewerkingen van de activerende instructie. Zowel de WHEN-component als de trigger-stepinstructies hebben toegang tot waarden uit de rij die wordt ingevoegd, verwijderd of bijgewerkt door middel van verwijzingen in de vorm van NEW.column-name en OLD.column-name, waarbij column-name de naam is van een kolom uit de tabel waaraan de trigger is gekoppeld. De beschikbaarheid van de tabelverwijzingen OLD en NEW is afhankelijk van het type database-event waarmee de trigger werkt:

- INSERT – NEW-verwijzingen zijn geldig
- UPDATE – NEW en OLD-verwijzingen zijn geldig
- DELETE – OLD-verwijzingen zijn geldig

De opgegeven tijdsbepaling (BEFORE, AFTER of INSTEAD OF) bepaalt wanneer de trigger-stepinstructies worden uitgevoerd ten opzichte van invoeging, wijziging of verwijdering van de gekoppelde rij. Een ON CONFLICT-component kan worden opgegeven als onderdeel van een instructie UPDATE of INSERT in een trigger-step. Als een ON CONFLICT-component echter wordt opgegeven als onderdeel van de instructie die ertoe leidt dat de trigger wordt geactiveerd, wordt in plaats daarvan het conflictoplossingsbeleid gebruikt.

Naast tabeltriggers kan een trigger van het type INSTEAD OF worden gemaakt voor een weergave. Als een of meer triggers van het type INSTEAD OF INSERT, INSTEAD OF DELETE of INSTEAD OF UPDATE worden gedefinieerd voor een weergave, wordt het niet als onjuist beschouwd om het bijbehorende type instructie (INSERT, DELETE of UPDATE) uit te voeren op de weergave. In dat geval leidt de uitvoering van een INSERT, DELETE of UPDATE op de weergave tot activering van de gekoppelde triggers. Aangezien de trigger een INSTEAD OF-trigger is, worden de tabellen die ten grondslag liggen aan de weergave, niet gewijzigd door de instructie die de activering van de trigger heeft veroorzaakt. De triggers kunnen echter worden gebruikt voor het uitvoeren van wijzigende bewerkingen op de onderliggende tabellen.

Als u een trigger voor een tabel met een kolom INTEGER PRIMARY KEY maakt, moet u rekening houden met het volgende. Als een trigger van het type BEFORE de kolom INTEGER PRIMARY KEY wijzigt van een rij die wordt bijgewerkt door de instructie die activering van de trigger heeft veroorzaakt, vindt er geen bijwerking plaats. Een oplossing is het maken van de tabel met een kolom PRIMARY KEY in plaats van een kolom INTEGER PRIMARY KEY.

Een trigger kan worden verwijderd met behulp van de instructie DROP TRIGGER. Wanneer een tabel of weergave wordt verwijderd, worden ook automatisch alle triggers verwijderd die zijn gekoppeld aan die tabel of weergave.

RAISE()-functie

Een speciale SQL-functie RAISE() kan worden gebruikt in een trigger-stepinstructie van een trigger. Deze functie heeft de volgende syntaxis:

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

Wanneer een van de eerste drie vormen wordt aangeroepen tijdens het uitvoeren van de trigger, wordt de opgegeven ON CONFLICT-verwerkingsactie (ABORT, FAIL of ROLLBACK) uitgevoerd en wordt het uitvoeren van de huidige instructie beëindigd. De ROLLBACK wordt beschouwd als een fout bij het uitvoeren van de instructie, waardoor de SQLStatement-instantie waarvan de methode execute() werd uitgevoerd, een gebeurtenis error (SQLErrorEvent.ERROR) verzendt. Van het SQLError-object in de eigenschap error van het verzonden gebeurtenisobject wordt de eigenschap details ingesteld op het error-message die is opgegeven in de functie RAISE().

Wanneer RAISE(IGNORE) wordt aangeroepen, worden het restant van de huidige trigger, de instructie die uitvoering van de trigger heeft veroorzaakt, en alle daaropvolgende triggers die zouden zijn uitgevoerd, afgebroken. Er worden geen databasewijzigingen teruggedraaid. Als de instructie die uitvoering van de trigger heeft veroorzaakt, zelf onderdeel van een trigger is, wordt de uitvoering van het triggerprogramma hervat bij het begin van de volgende stap. Zie het gedeelte ON CONFLICT (conflictalgoritmen) voor meer informatie over algoritmen voor het oplossen van conflicten.

DROP TRIGGER

De instructie DROP TRIGGER verwijdert een trigger die is gemaakt met de instructie CREATE TRIGGER.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

De trigger wordt verwijderd uit de database. Denk eraan dat triggers automatisch worden verwijderd wanneer de tabel waaraan ze zijn gekoppeld wordt verwijderd.

Speciale instructies en componenten

In dit gedeelte worden diverse componenten beschreven die een uitbreiding vormen op SQL zoals voorhanden in runtime, alsmede twee taalelementen die kunnen worden gebruikt in vele instructies, opmerkingen en expressies.

COLLATE

De component COLLATE wordt gebruikt in instructies van het type SELECT, CREATE TABLE en CREATE INDEX om het vergelijkingsalgoritme op te geven dat wordt gebruikt bij het vergelijken of sorteren van waarden.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

Het standaardsorteertype voor kolommen is BINARY. Wanneer de sortering BINARY wordt gebruikt bij waarden van de opslagklasse TEXT, wordt binaire sortering uitgevoerd door de bytes in het geheugen die de waarde vertegenwoordigen te vergelijken ongeacht de tekstcodering.

De sorteervolgorde NOCASE wordt alleen toegepast op waarden van de opslagklasse TEXT. Bij gebruik wordt bij de sortering NOCASE geen onderscheid gemaakt tussen hoofdletters en kleine letters.

Er wordt geen sorteervolgorde gebruik voor opslagklassen van het type NULL, BLOB, INTEGER of REAL.

Voor het gebruik van een ander sorteertype dan BINARY bij een kolom moet een COLLATE-component worden opgegeven als onderdeel van de kolomdefinitie in de instructie CREATE TABLE. Wanneer twee TEXT-waarden worden vergeleken, wordt een sorteervolgorde gebruikt om in overeenstemming met de volgende regels de resultaten van de vergelijking vast te stellen:

- Bij binaire vergelijkingsoperatoren geldt dat, als een van beide operanden een kolom is, het standaardsorteertype van de kolom de sorteervolgorde bepaalt die wordt gebruik voor de vergelijking. Als beide operanden kolommen zijn, bepaalt het sorteertype voor de linkeroperand de gebruikte sorteervolgorde. Als geen van beide operanden een kolom is, wordt de sorteervolgorde BINARY gebruikt.
- De operator BETWEEN...AND is gelijk aan het gebruik van twee expressies met de operatoren >= en <=. Zo is de expressie x BETWEEN y AND z equivalent aan x >= y AND x <= z. Daarom volgt de operator BETWEEN...AND de voorafgaande regel voor het bepalen van de sorteervolgorde.
- De operator IN gedraagt zich als de operator = met als doel de te gebruiken sorteervolgorde te bepalen. Zo is de sorteervolgorde die wordt gebruikt voor de expressie x IN (y, z), het standaardsorteertype van x als x een kolom is. Anders wordt de sortering BINARY gebruikt.
- Aan een ORDER BY-component die onderdeel is van een SELECT-instructie, kan expliciet een sorteervolgorde worden toegewezen die moet worden gebruikt voor de sorteerbewerking. In dat geval wordt altijd de expliciete sorteervolgorde gebruikt. Als de door een ORDER BY-component gesorteerde expressie een kolom is, wordt anders het standaardsorteertype van de kolom gebruikt om de sorteervolgorde te bepalen. Als de expressie geen kolom is, wordt de sorteervolgorde BINARY gebruikt.

EXPLAIN

De opdrachtmodifier EXPLAIN is een niet-standaard uitbreiding op SQL.

```
sql-statement  ::=  EXPLAIN sql-statement
```

Als het trefwoord EXPLAIN voorkomt vóór een andere SQL-instructie, wordt de instructie feitelijk niet uitgevoerd, maar meldt het resultaat de volgorde van virtuele machine-instructies die zouden zijn gebruikt bij de uitvoering van die instructie, als het trefwoord EXPLAIN niet aanwezig was geweest. De functie EXPLAIN is een geavanceerde functie waarmee ontwikkelaars tekst in SQL-instructies kunnen wijzigen voor het optimaliseren van prestaties of voor foutopsporing in een instructie die niet goed lijkt te functioneren.

ON CONFLICT (conflictalgoritmen)

De component ON CONFLICT is geen afzonderlijke SQL-instructie. Het betreft een niet-standaard component die in vele andere SQL-instructies kan voorkomen.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

De eerste vorm van de component ON CONFLICT, die de trefwoorden ON CONFLICT gebruikt, wordt gebruikt in een instructie CREATE TABLE. Voor een instructie INSERT of UPDATE wordt de tweede vorm gebruikt, waarbij ON CONFLICT wordt vervangen door OR om de syntaxis natuurlijker te doen lijken. In plaats van INSERT ON CONFLICT IGNORE wordt de instructie bijvoorbeeld INSERT OR IGNORE. Hoewel de trefwoorden anders zijn, is de betekenis van de component in beide gevallen gelijk.

De component ON CONFLICT bepaalt welk algoritme wordt gebruikt voor het oplossen van beperkingsconflicten. De vijf algoritmen zijn ROLLBACK, ABORT, FAIL, IGNORE en REPLACE. Het standaardalgoritme is ABORT. Hierna volgt een beschrijving van de vijf conflictalgoritmen:

ROLLBACK Wanneer er een schending van een beperking optreedt, leidt dit tot een onmiddellijke ROLLBACK en wordt de huidige transactie beëindigd. De instructie wordt afgebroken en de SQLStatement-instantie verzendt een error-gebeurtenis. Als er geen transactie actief is (behalve de geïmpliceerde transactie die bij elke instructie wordt gemaakt), werkt dit algoritme hetzelfde als ABORT.

ABORT Wanneer er een schending van een beperking optreedt, maakt de instructie alle eventuele, reeds aangebrachte wijzigingen ongedaan en verzendt de SQLStatement-instantie een error-gebeurtenis. Er wordt geen ROLLBACK uitgevoerd, waardoor de wijzigingen van eerdere instructies binnen een transactie behouden blijven. ABORT is het standaardgedrag.

FAIL Wanneer er een schending van een beperking optreedt, wordt de instructie afgebroken en verzendt de SQLStatement-instantie een error-gebeurtenis. Alle wijzigingen aan de database die de instructie heeft aangebracht voordat de schending van de beperking zich voordeed, blijven echter behouden en worden niet ongedaan gemaakt. Als een UPDATE-instructie bijvoorbeeld een schending van een beperking opmerkt in de 100e rij die deze probeert bij te werken, blijven de wijzigingen in de eerste 99 rijen behouden, maar vinden er geen wijzigingen in rij 100 en daaropvolgende rijen plaats.

IGNORE Wanneer er een schending van een beperking optreedt, wordt die ene rij waarin de schending van een beperking zich voordeed, niet ingevoegd of gewijzigd. Met uitzondering van het negeren van deze rij wordt de rest van de instructie normaal uitgevoerd. Andere rijen vóór en na de rij met de schending van de beperking worden normaal ingevoegd of bijgewerkt. Er wordt geen fout geretourneerd. Er wordt geen fout geretourneerd.

REPLACE Wanneer er een schending UNIQUE van een beperking optreedt, worden de reeds bestaande rijen die de schending van de beperking veroorzaken, eerst verwijderd voordat de huidige wordt ingevoegd of bijgewerkt. Hierdoor vindt het invoegen of bijwerken altijd plaats en wordt de instructie normaal uitgevoerd. Er wordt geen fout geretourneerd. Als zich een schending van een NOT NULL-beperking voordoet, wordt de waarde NULL vervangen door de standaardwaarde voor die kolom. Als de kolom geen standaardwaarde heeft, wordt het algoritme ABORT gebruikt. Als zich een schending van een CHECK-beperking voordoet, wordt het algoritme IGNORE gebruikt. Wanneer deze conflictoplossingsstrategie rijen verwijdert om aan een beperking te voldoen, worden voor deze rijen geen triggers voor verwijderen geactiveerd.

Het algoritme dat is opgegeven in de OR-component van een INSERT- of UPDATE-instructie heeft voorrang op algoritmen die zijn opgegeven in een CREATE TABLE-instructie. Als er geen algoritme is opgegeven in de instructie CREATE TABLE of de uitvoerende instructie INSERT of UPDATE, wordt het algoritme ABORT gebruikt.

REINDEX

De instructie REINDEX wordt gebruikt voor het verwijderen en opnieuw maken van een of meer indices. Deze instructie is handig wanneer de definitie van een sorteervolgorde is gewijzigd.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

Bij de eerste vorm worden alle indices in alle gekoppelde databases die de genoemde sorteervolgorde gebruiken, opnieuw gemaakt. Bij de tweede vorm, wanneer een table-name wordt opgegeven, worden alle indices die aan de tabel zijn gekoppeld, opnieuw samengesteld. Als er een index-name is opgegeven, wordt alleen de opgegeven index verwijderd en opnieuw gemaakt.

OPMERKINGEN

Opmerkingen zijn geen SQL-instructies, maar kunnen wel voorkomen in SQL-query's. Ze worden door de runtime als witruimte beschouwd. Ze kunnen beginnen overal waar witruimte kan worden gevonden, inclusief binnen expressies die meerdere regels beslaan.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

Een opmerking van één regel wordt aangegeven met twee liggende streepjes. Een opmerking van één regel loopt slechts tot het einde van de huidige regel.

Blokopmerkingen kunnen een willekeurig aantal regels beslaan of kunnen zijn ingebed binnen één regel. Als er geen beëindigend scheidingsteken is, loopt een blokopmerking door tot aan het einde van de invoer. Deze situatie wordt niet als een fout gezien. Een nieuwe SQL-instructie kan beginnen op een regel nadat een blokopmerking is geëindigd. Blokopmerkingen kunnen overal waar witruimte kan voorkomen worden ingesloten, inclusief binnen expressies en te midden van andere SQL-instructies. Blokopmerkingen kunnen niet worden genest. Opmerkingen van één regel binnen een blokopmerking worden genegeerd.

EXPRESSIES

Expressies zijn subopdrachten binnen andere SQL-blokken. Hierna volgt een beschrijving van de geldige syntaxis voor een expressie binnen een SQL-instructie:

```
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
```

Een expressie is een willekeurige combinatie van waarden en operatoren die resulteert in één waarde. Expressies kunnen worden onderverdeeld in twee algemene typen, op basis van of ze resulteren in een booleaanse waarde (waar of onwaar), of in een niet-booleaanse waarde.

In verscheidene veelvoorkomende situaties, inclusief in een WHERE-component, een HAVING-component, de expressie ON in een JOIN-component of een CHECK-expressie, moet de expressie resulteren in een booleaanse waarde. De volgende typen expressies voldoen aan deze voorwaarde:

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- Bepaalde functies
- Bepaalde operatoren (in het bijzonder vergelijkingsoperatoren)

Letterlijke waarden
Een letterlijke numeriek waarde wordt geschreven is als geheel getal of als een zwevende-kommagetal. Wetenschappelijke notatie wordt ondersteund. De . (punt) wordt altijd gebruikt als decimaalteken.

Een letterlijke tekenreeks wordt aangeduid door de tekenreeks tussen enkele aanhalingstekens ' te zetten. Als u een enkel aanhalingsteken in een tekenreeks wilt opnemen, plaatst u twee enkele aanhalingstekens achter elkaar ('').

Een booleaanse letterlijke waarde wordt aangeduid door de waarde true of false. Letterlijke booleaanse waarden worden gebruikt bij het kolomgegevenstype Booleaans.

Een BLOB is een letterlijke tekenreeks met hexadecimale gegevens, die wordt voorafgegaan door één teken x of X, zoals X'53514697465'.

Een letterlijke waarde kan ook bestaan uit het token NULL.

Kolomnaam
Een kolomnaam kan elk van de namen zijn, die zijn gedefinieerd in de instructie CREATE TABLE, of een van de volgende speciale ID's: ROWID, OID, of _ROWID_. Deze speciale ID's beschrijven alle de unieke willekeurige sleutel van gehele getallen (de rijsleutel) die is gekoppeld aan elke rij van elke tabel. De speciale ID's verwijzen alleen naar de rijsleutel als de instructie CREATE TABLE geen echte kolom met dezelfde naam definieert. Rijsleutels gedragen zich als alleen-lezen kolommen. Een rijsleutel kan worden gebruikt overal waar een standaardkolom kan worden gebruikt, met deze uitzondering dat u de waarde van een rijsleutel niet kunt wijzigen in een UPDATE- of INSERT-instructie. De instructie SELECT * FROM table bevat geen rijsleutel in de bijbehorende resultatenset.

Instructie SELECT
Een SELECT-instructie kan voorkomen in een expressie als rechteroperand van de operator IN, als scalaire hoeveelheid (één resultaatwaarde) of als operand van een EXISTS-operator. Bij gebruik als scalaire hoeveelheid of als operand van een IN-operator kan de SELECT slechts één kolom in het resultaat hebben. Een samengestelde SELECT-instructie (die is verbonden door trefwoorden zoals UNION of EXCEPT) is toegestaan. Met de operator EXISTS worden de kolommen in de resultatenset van de SELECT genegeerd en retourneert de expressie TRUE als er een of

meer rijen zijn, en FALSE als de resultatenset leeg is. Als er geen termen in de SELECT-expressie verwijzen naar de waarde in de bevattende query, wordt de expressie één maal geëvalueerd voordat er verdere verwerking plaatsvindt, en wordt het resultaat desnoods opnieuw gebruikt. Als de expressie SELECT geen variabelen bevat van de buitenste query (beter bekend als een gecorreleerde subquery), wordt de SELECT telkens wanneer dit nodig is, opnieuw geëvalueerd.

Wanneer een SELECT de rechteroperand van de operator IN is, geeft de operator IN TRUE als resultaat als het resultaat van de linkeroperand gelijk is aan een van de waarden in de resultatenset van de instructie SELECT. De operator IN kan worden voorafgegaan door het trefwoord NOT om de bedoeling van de test om te keren.

Wanneer een SELECT wel voorkomt binnen een expressie, maar niet als rechteroperand van een IN-operator, wordt de eerst rij van het resultaat van de SELECT de waarde die wordt gebruikt in de expressie. Als de SELECT meer dan één resultaatrij geeft, worden alle rijen na de eerste rij genegeerd. Als de SELECT geen rijen geeft, is de waarde van de SELECT NULL.

Expressie CAST

Een CAST-expressie wijzigt het opgegeven gegevenstype in het gegevenstype dat is gegeven. Het opgegeven type kan elke niet-lege typenaam zijn die geldig is voor het type in een kolomdefinitie van een CREATE TABLE-instructie. Zie Ondersteuning van gegevenstypen voor meer informatie.

Aanvullende expressie-elementen

De volgende SQL-elementen kunnen ook worden gebruikt in expressies:

- Ingebouwde functies: statistische functies, scalaire functies en datum- en tijdnotatiefuncties
- Operatoren
- Parameters

Ingebouwde functies

De ingebouwde functies bestaan uit drie hoofdcategorieën:

- Statistische functies
- Scalaire functies
- Datum- en tijdnotatiefuncties

Naast deze functies is er een speciale functie RAISE() die wordt gebruikt voor het melden van een fout bij het uitvoeren van een trigger. Deze functie kan alleen worden gebruikt binnen de hoofdtekst van een CREATE TRIGGER-instructie. Zie CREATE TRIGGER > RAISE() voor meer informatie over de functie RAISE().

Evenals alle trefwoorden in SQL zijn functienamen niet hoofdlettergevoelig.

Statistische functies

Statistische functies voeren bewerkingen uit op waarden uit meerdere rijen. Deze functies worden voornamelijk gebruikt in SELECT-instructies in combinatie met de component GROUP BY.

AVG(X)	Retourneert de gemiddelde waarde van alle niet-NULL X binnen een groep. Tekenreeks- en BLOB-waarden die niet op getallen lijken, worden geïnterpreteerd als 0. Het resultaat van AVG() is altijd een zwevende-kommawaarde, zelfs als alle invoer bestaat uit gehele getallen.
COUNT(X)	De eerste vorm retourneert een telling van het aantal malen dat X niet NULL is in een groep.
COUNT(*)	De tweede vorm (met het argument *) retourneert het totaalaantal rijen in de groep.
MAX(X)	Retourneert de maximumwaarde van alle waarden in de groep. Voor het bepalen van het maximum wordt de gebruikelijke sorteervolgorde gebruikt.

MIN(X)	Retourneert de minimale niet-NULL-waarde van alle waarden in de groep. Voor het bepalen van het minimum wordt de gebruikelijke sorteervolgorde gebruikt. Als alle waarden in de groep NULL zijn, wordt NULL geretourneerd.
SUM(X)	Retourneert de numerieke som van alle niet-NULL-waarden in de groep. Als alle waarden NULL zijn, retourneert SUM() NULL, en retourneert TOTAL() 0.0. Het resultaat van TOTAL() is altijd een zwevende-kommawaarde. Het resultaat van SUM() is een geheel-getalwaarde als alle niet-NULL-invoer uit gehele getallen bestaat. Als een willekeurige invoer voor SUM() niet een geheel getal en niet-NULL is, retourneert SUM() een zwevende-kommawaarde. Deze waarde kan een benadering zijn van de werkelijke som.
TOTAL(X)	

In elke van de voorafgaande statistische functies met één argument kan dat argument worden voorafgegaan door het trefwoord DISTINCT. In dat geval worden dubbele elementen gefilterd voordat ze worden doorgegeven aan de statistische functie. Zo retourneert de functieaanroep COUNT(DISTINCT x) het aantal afzonderlijke waarden in kolom X, in plaats van het totaalaantal niet-NULL-waarden in kolom x.

Scalaire functies

Scalaire functies bewerken waarden met één rij per keer.

ABS(X)	Retourneert de absolute waarde van argument X.
COALESCE(X, Y, ...)	Retourneert een kopie van het eerste niet-NULL-argument. Als alle argumenten NULL zijn, wordt NULL geretourneerd. Er moeten ten minste twee argumenten zijn.
GLOB(X, Y)	Deze functie wordt gebruikt voor het implementeren van de syntaxis X GLOB Y.
IFNULL(X, Y)	Retourneert een kopie van het eerste niet-NULL-argument. Als beide argumenten NULL zijn, wordt NULL geretourneerd. Deze functie gedraagt zich hetzelfde als COALESCE().
HEX(X)	Het argument wordt geïnterpreteerd als een waarde van het BLOB-opslagtype. Het resultaat is een hexadecimale weergave van de inhoud van die waarde.
LAST_INSERT_ROWID( )	Retourneert de rij-ID (gegenereerde primaire sleutel) van de laatste rij die is ingevoegd in de database via de huidige SQLConnection. Deze waarde is identiek aan de waarde die wordt geretourneerd door de eigenschap SQLConnection.lastInsertRowID.
LENGTH(X)	Retourneert de tekenreekslengte van X in lettertekens.
LIKE(X, Y [, Z])	Deze functie wordt gebruikt voor het implementeren van de syntaxis X LIKE Y [ESCAPE Z] van SQL. Als de optionele component ESCAPE aanwezig is, wordt de functie aangeroepen met drie argumenten. Anders wordt deze aangeroepen met slechts twee argumenten.
LOWER(X)	Retourneert een kopie van tekenreeks X met alle lettertekens omgezet naar kleine letters.
LTRIM(X) LTRIM(X, Y)	Retourneert een tekenreeks die wordt gevormd door het verwijderen van spaties vanaf de linkerzijde van X. Als een argument Y is opgegeven, verwijdert de functie alle lettertekens in Y vanaf de linkerzijde van X.
MAX(X, Y, ...)	Retourneert het argument met de maximumwaarde. Argumenten kunnen behalve getallen ook tekenreeksen zijn. De maximumwaarde wordt bepaald door de gedefinieerde sorteervolgorde. Denk eraan dat MAX() een enkelvoudige functie is wanneer deze 2 of meer argumenten heeft, maar een statistische functie wanneer deze één argument heeft.
MIN(X, Y, ...)	Retourneert het argument met de minimumwaarde. Argumenten kunnen behalve getallen ook tekenreeksen zijn. De minimumwaarde wordt bepaald door de gedefinieerde sorteervolgorde. Denk eraan dat MIN() een enkelvoudige functie is wanneer deze 2 of meer argumenten heeft, maar een statistische functie wanneer deze één argument heeft.
NULLIF(X, Y)	Retourneert het eerste argument als de argumenten verschillend zijn. Retourneert NULL als de argumenten gelijk zijn.
QUOTE(X)	Deze routine retourneert een tekenreeks die de waarde is van het argument ervan, dat geschikt is voor opname in een andere SQL-instructie. Tekenreeksen worden omgeven door enkele aanhalingstekens met desnoods escape-tekens voor inwendige aanhalingstekens. BLOB-opslagklassen worden gecodeerd als hexadecimale letterlijke waarden. De functie is handig bij het schrijven van triggers voor de implementatie van de functie voor ongedaan maken en opnieuw uitvoeren.
RANDOM(*)	Retourneert een pseudowillekeurig geheel getal tussen -9223372036854775808 en 9223372036854775807. Deze willekeurige waarde is niet erg geschikt voor cryptografische doeleinden.
RANDOMBLOB(N)	Retourneert een N-byte BLOB met pseudowillekeurige bytes. N moet een positief geheel getal zijn. Deze willekeurige waarde is niet erg geschikt voor cryptografische doeleinden. Als de waarde van N negatief is, wordt één byte geretourneerd.
ROUND(X) ROUND(X, Y)	Rondt het getal X af naar Y cijfers achter het decimaalteken. Als het argument Y wordt weggelaten, wordt 0 gebruikt.

RTRIM(X) RTRIM(X, Y)	Retourneert een tekenreeks die wordt gevormd door het verwijderen van spaties vanaf de rechterzijde van X. Als een argument Y is opgegeven, verwijdert de functie alle lettertekens in Y vanaf de rechterzijde van X.
SUBSTR(X, Y, Z)	Retourneert een subtekenreeks van de invoertekenreeks X die begint met het Y-de teken en die Z tekens lang is. Het letterteken helemaal links van X heeft indexpositie 1. Als Y negatief is, wordt het eerste letterteken van de subtekenreeks gevonden door te tellen vanaf rechts, in plaats van links.
TRIM(X) TRIM(X, Y)	Retourneert een tekenreeks die wordt gevormd door het verwijderen van spaties vanaf de rechterzijde van X. Als een argument Y is opgegeven, verwijdert de functie alle lettertekens in Y vanaf de rechterzijde van X.
TYPEOF(X)	Retourneert het type van de expressie X. De mogelijke retourwaarden zijn 'null', 'integer', 'real', 'text' en 'blob'. Zie Ondersteuning van gegevenstypen voor meer informatie over gegevenstypen.
UPPER(X)	Retourneert een kopie van de invoertekenreeks X die in zijn geheel is omgezet in hoofdletters.
ZEROBLOB(N)	Retourneert een BLOB met N bytes van 0x00.

Datum- en tijdnotatiefuncties

De datum- en tijdnotatiefuncties vormen een groep scalaire functies die worden gebruikt voor het maken van opgemaakte datum- en tijdgegevens. Houd er rekening mee dat deze functies tekenreeks- en getalwaarden bewerken en retourneren. Deze functies zijn niet bedoeld voor gebruik in combinatie met het gegevenstype DATE. Bij gebruik van deze functies voor gegevens in een kolom waarvan het gedeclareerde gegevenstype DATE is, gedragen zij zich niet zoals verwacht.

DATE(T, ...)	De functie DATE() retourneert een tekenreeks met de datum in de notatie YYYY-MM-DD. De eerste parameter (T) geeft een tijdtekenreeks aan in de notatie die kan worden gevonden onder Tijdnotaties. Na de tijdtekenreeks kan een willekeurig aantal modifiers worden opgegeven. De modifiers kunt u terugvinden onder Modifiers.
TIME(T, ...)	De functie TIME() retourneert een tekenreeks met daarin de tijd als HH:MM:SS. De eerste parameter (T) geeft een tijdtekenreeks aan in de notatie die kan worden gevonden onder Tijdnotaties. Na de tijdtekenreeks kan een willekeurig aantal modifiers worden opgegeven. De modifiers kunt u terugvinden onder Modifiers.
DATETIME(T, ...)	De functie DATETIME() retourneert een tekenreeks met daarin de datum en tijd met de notatie YYYY-MM-DD HH:MM:SS. De eerste parameter (T) geeft een tijdtekenreeks aan in de notatie die kan worden gevonden onder Tijdnotaties. Na de tijdtekenreeks kan een willekeurig aantal modifiers worden opgegeven. De modifiers kunt u terugvinden onder Modifiers.
JULIANDAY(T, ...)	De functie JULIANDAY() retourneert een getal dat het aantal dagen aangeeft sinds 12 uur 's middags in Greenwich op 24 november 4714 v. Chr. alsmede de geleverde datum. De eerste parameter (T) geeft een tijdtekenreeks aan in de notatie die kan worden gevonden onder Tijdnotaties. Na de tijdtekenreeks kan een willekeurig aantal modifiers worden opgegeven. De modifiers kunt u terugvinden onder Modifiers.

STRFTIME(F, T, ...)	De routine STRFTIME() retourneert de datum met een notatie volgens de notatietekenreeks die is opgegeven als het eerste argument F. De notatietekenreeks ondersteunt de volgende substituties: %d - dag van de maand %f - fractionele seconden SS.SSS %H - uur 00-24 %j - dag van het jaar 001-366 %J - Juliaans dagnummer %m - maand 01-12 %M - minuut 00-59 %s - seconden sinds 01-01-1970 %S - seconden 00-59 %w - dag van de week 0-6 (zondag = 0) %W - week van het jaar 00-53 %Y - jaar 0000-9999 %% - % De tweede parameter (T) geeft een tijdtekenreeks aan in de notatie die kan worden gevonden onder Tijdnotaties. Na de tijdtekenreeks kan een willekeurig aantal modifiers worden opgegeven. De modifiers kunt u terugvinden onder Modifiers.

Tijdnotaties

Een tijdtekenreeks kan een van de volgende notaties hebben:

YYYY-MM-DD	2007-06-15
YYYY-MM-DD HH:MM	2007-06-15 07:30
YYYY-MM-DD HH:MM:SS	2007-06-15 07:30:59
YYYY-MM-DD HH:MM:SS.SSS	2007-06-15 07:30:59.152
YYYY-MM-DDTHH:MM	2007-06-15T07:30
YYYY-MM-DDTHH:MM:SS	2007-06-15T07:30:59
YYYY-MM-DDTHH:MM:SS.SSS	2007-06-15T07:30:59.152
HH:MM	07:30 (datum is 01-01-2000)
HH:MM:SS	07:30:59 (datum is 01-01-2000)
HH:MM:SS.SSS	07:30:59:152 (datum is 01-01-2000)
now	Huidige datum en tijd in Universal Coordinated Time.
DDDD.DDDD	Juliaans dagnummer

Het teken T in deze notaties is een letterteken "T" dat de datum en de tijd scheidt. Notaties met daarin alleen een tijd gaan uit van de datum 01-01-2001.

Modifiers

De tijdtekenreeks kan worden gevolgd door nul of meer modifiers die de datum wijzigen dan wel de interpretatie van de datum wijzigen. De volgende modifiers zijn beschikbaar:

NNN days	Aantal dagen dat moet worden toegevoegd aan de tijd.
NNN hours	Aantal uren dat moet worden toegevoegd aan de tijd.
NNN minutes	Aantal minuten dat moet worden toegevoegd aan de tijd.
NNN.NNNN seconds	Aantal seconden en milliseconden dat moet worden toegevoegd aan de tijd.
NNN months	Aantal maanden dat moet worden toegevoegd aan de tijd.
NNN years	Aantal jaren dat moet worden toegevoegd aan de tijd.
start of month	Ga terug in de tijd tot het begin van de maand.

start of year	Ga terug in de tijd tot het begin van het jaar.
start of day	Ga terug in de tijd tot het begin van de dag.
weekday N	Ga vooruit in de tijd naar de opgegeven weekdag. (0 = zondag, 1 = maandag, enzovoort)
localtime	Converteert de datum naar lokale tijd.
utc	Converteert de datum naar Universal Coordinated Time

Operatoren

SQL ondersteunt een grote verscheidenheid aan operatoren, inclusief veel gebruikte operatoren die in de meeste programmeertalen voorkomen, alsmede diverse operatoren die uniek zijn voor SQL.

Algemene operatoren

De volgende binaire operatoren zijn toegestaan in een SQL-blok en worden vermeld in de volgorde van hoogste naar laagste voorrang:

```
*     /     %
+     -
<< >> & |
< >=    > >=
=     ==    !=    <> IN
AND
OR
```

Ondersteunde unaire prefixoperatoren (voorvoegseloperatoren) zijn:

```
 !     ~     NOT
```

De operator COLLATE kan worden beschouwd als een unaire postfixoperator (achtervoegseloperator). De operator COLLATE heeft de hoogste voorrang. Deze bindt altijd hechter dan elke andere unaire prefixoperator of elke andere binaire operator.

Denk eraan dat er twee varianten zijn van de operatoren equals (is gelijk aan) en unequals (is niet gelijk aan). Equals (is gelijk aan) kan = of == zijn. De operator not equals (is niet gelijk aan) kan != of <> zijn.

De operator || is een tekenreekssamenvoegingsoperator: deze voegt twee tekenreeksen van zijn operanden samen.

De operator % geeft als uitvoer het restant van de linkeroperand modulo de rechteroperand.

Het resultaat van elke binaire operator is een numerieke waarde, behalve voor de samenvoegingsoperator || die een tekenreeksresultaat geeft.

SQL-operatoren

LIKE

De operator LIKE voert een vergelijking van patroonovereenkomsten uit.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

De operand rechts van de operator LIKE bevat het patroon en de linkeroperator bevat de tekenreeks die met het patroon moet worden vergeleken. Een percentageteken (%) in het patroon is een jokerteken: het komt overeen met een willekeurige reeks van nul of meer tekens in de tekenreeks. Een onderstrepingsteken (_) in het patroon komt overeen met één willekeurig letterteken in de tekenreeks. Elk ander letterteken komt overeen met zichzelf of zijn equivalent in hoofdletter/kleine letter. Dat betekent dat geen onderscheid wordt gemaakt tussen hoofdletters en kleine

letters bij het zoeken naar overeenkomsten. (Opmerking: de database-engine kan alleen hoofdletters/kleine letters voor 7-bits Latijnse lettertekens interpreteren. Daarom is de operator LIKE hoofdlettergevoelig voor 8-bits, iso-8859-lettertekens of UTF-8-lettertekens. Zo geeft de expressie 'a' LIKE 'A' als resultaat TRUE, maar geeft 'æ' LIKE 'Æ' als resultaat FALSE.) Hoofdlettergevoeligheid voor Latijnse lettertekens kan worden gewijzigd met behulp van de eigenschap SQLConnection.caseSensitiveLike.

Als de optionele component ESCAPE aanwezig is, moet de expressie die volgt op het trefwoord ESCAPE, worden geëvalueerd met een tekenreeks die uit één letterteken bestaat. Dit letterteken kan worden gebruikt in het patroon LIKE voor het zoeken naar overeenkomsten met letterlijke percentage- of onderstrepingstekens. Het escape-teken gevolgd door een percentageteken, onderstrepingsteken of zichzelf komt overeen met respectievelijk een letterlijk percentageteken, onderstrepingsteken of escape-teken in de tekenreeks.

GLOB

De operator GLOB komt overeen met LIKE, maar gebruikt voor zijn jokertekens de globbingsyntaxis voor Unix-bestanden. In tegenstelling tot LIKE is GLOB wel hoofdlettergevoelig.

IN

De operator IN berekent of de linkeroperand gelijk is aan een van de waarden in de rechteroperand (een reeks waarden tussen haakjes).

```
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]*
```

De rechteroperand kan bestaan uit een reeks met komma's gescheiden letterlijke waarden of kan het resultaat zijn van een SELECT-instructie. Zie de SELECT-instructies in expressies voor een uitleg en de beperkingen bij het gebruik van de instructie SELECT als rechteroperand van de operator IN.

BETWEEN...AND

De operator BETWEEN...AND is gelijk aan het gebruik van twee expressies met de operatoren >= en <=. Zo is de expressie x BETWEEN y AND z equivalent aan x >= y AND x <= z.

NOT

De operator NOT is een ontkenningsoperator. De operatoren GLOB, LIKE en IN kunnen worden voorafgegaan door het trefwoord NOT om de betekenis van de test om te keren (dat wil zeggen om te controleren of een waarde niet overeenkomt met het opgegeven patroon).

Parameters

Een parameter geeft een tijdelijke aanduiding in de expressie aan voor een letterlijke waarde, die in runtime wordt vervangen door een waarde toe te kennen aan de gekoppelde array SQLStatement.parameters. Parameters kunnen drie vormen hebben:

?	Een vraagteken geeft een geïndexeerde parameter aan. Parameters krijgen numerieke (op nul gebaseerde) indexwaarden toegewezen op basis van de volgorde in de instructie.
:AAAA	Een dubbele punt die wordt gevolgd door de naam van een ID, houdt een plaats gereserveerd voor een benoemde parameter met de naam AAAA. Benoemde parameters worden ook genummerd op basis van de volgorde in de SQL-instructie. U kunt verwarring vermijden door benoemde en genummerde parameters door elkaar te gebruiken.
@AAAA	Een apestaartje is equivalent aan een dubbele punt.

Niet-ondersteunde SQL-functies

Hierna volgt een lijst met de standaard SQL-elementen die niet worden ondersteund in Adobe AIR:

FOREIGN KEY-beperkingen FOREIGN KEY-beperkingen worden wel geparseerd (geanalyseerd), maar niet afgedwongen.

Triggers FOR EACH STATEMENT-triggers worden niet ondersteund (alle triggers moeten FOR EACH ROW zijn). INSTEAD OF-triggers worden niet ondersteund voor tabellen (INSTEAD OF-triggers zijn alleen toegestaan voor weergaven). Recursieve triggers (triggers die zichzelf activeren) worden niet ondersteund.

ALTER TABLE Alleen de varianten RENAME TABLE en ADD COLUMN van de instructie ALTER TABLE worden ondersteund. Andere soorten ALTER TABLE-bewerkingen, zoals DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT enzovoort worden genegeerd.

Geneste transacties Er wordt slechts één actieve transactie toegestaan.

RIGHT en FULL OUTER JOIN RIGHT OUTER JOIN en FULL OUTER JOIN worden niet ondersteund.

Bijwerkbare VIEW Een weergave is alleen-lezen. U kunt geen instructie van het type DELETE, INSERT of UPDATE uitvoeren op een weergave. Een INSTEAD OF-trigger die wordt geactiveerd bij een poging tot DELETE, INSERT of UPDATE van een weergave, wordt wel ondersteund en kan worden gebruikt voor het bijwerken van ondersteunende tabellen in de hoofdtekst van de trigger.

GRANT en REVOKE Een database is een normaal schijfbestand. De enige toegangsmachtigingen die kunnen worden toegepast, zijn de normale machtigingen voor bestandstoegang van het onderliggende besturingssysteem. De instructies GRANT en REVOKE die men vaak tegenkomt bij client-/server-RDBMS-en, worden niet geïmplementeerd.

De volgende SQL-elementen en SQLite-functies worden wel ondersteund in bepaalde SQLite-implementaties, maar worden niet ondersteund in Adobe AIR. De meeste van deze functies zijn beschikbaar via methoden van de klasse SQLConnection:

Aan transacties gerelateerde SQL-elementen (BEGIN, END, COMMIT, ROLLBACK) Deze functionaliteit is beschikbaar via de aan transacties gerelateerde methoden van de SQLConnection-klasse: SQLConnection.begin(), SQLConnection.commit() en SQLConnection.rollback().

ANALYZE Deze functionaliteit is beschikbaar via de methode SQLConnection.analyze().

ATTACH Deze functionaliteit is beschikbaar via de methode SQLConnection.attach().

COPY Deze instructie wordt niet ondersteund.

CREATE VIRTUAL TABLE Deze instructie wordt niet ondersteund.

DETACH Deze functionaliteit is beschikbaar via de methode SQLConnection.detach().

PRAGMA Deze instructie wordt niet ondersteund.

VACUUM Deze functionaliteit is beschikbaar via de methode SQLConnection.compact().

Toegang tot de systeemtabellen is niet beschikbaar. De systeemtabellen inclusief sqlite_master en andere tabellen met het prefix 'sqlite_' zijn niet beschikbaar in SQL-instructies. De runtime omvat een schema-API die een objectgeoriënteerde manier biedt voor toegang tot schemagegevens. Zie de methode SQLConnection.loadSchema() voor meer informatie.

Gewone-expressiefuncties (MATCH() en REGEX() Deze functies zijn niet beschikbaar in SQL-instructies.

De volgende functionaliteit is in vele SQLite-implementaties anders dan in Adobe AIR:

Geïndexeerde instructieparameters In veel implementaties wordt bij het indexeren van instructieparameters uitgegaan van één. In Adobe AIR zijn geďndexeerde instructieparameters echter gebaseerd op nul. Dit betekent dat de eerste parameter als index 0 krijgt, de tweede parameter de index 1 enzovoort.

INTEGER PRIMARY KEY-kolomdefinities Bij veel implementaties worden alleen kolommen die exact zijn gedefinieerd als INTEGER PRIMARY KEY gebruikt als de werkelijke primaire-sleutelkolom voor een tabel. Als u bij deze implementaties een ander gegevenstype gebruikt dat gewoonlijk een synoniem is voor INTEGER (zoals int), wordt de kolom niet gebruikt als de interne primaire sleutel. In Adobe AIR wordt het gegevenstype int (en andere INTEGER-synoniemen) beschouwd als precies equivalent aan INTEGER. Daarom wordt een kolom die is gedefinieerd als int PRIMARY KEY gebruikt als de interne primaire sleutel voor een tabel. Voor meer informatie gaat u naar de gedeelten CREATE TABLE en Kolomaffiniteit.

Aanvullende SQL-functies

De volgende kolomaffiniteitstypen worden standaard niet ondersteund in SQLite, maar worden wel ondersteund in Adobe AIR (evenals alle trefwoorden in SQL zijn deze namen voor gegevenstypen niet hoofdlettergevoelig):

Boolean komt overeen met de klasse Boolean.

Date komt overeen met de klasse Date.

int komt overeen met de klasse int (equivalent aan de kolomaffiniteit INTEGER).

Number komt overeen met de klasse Number (equivalent aan de kolomaffiniteit REAL).

Object komt overeen met de klasse Object of elke andere subklasse die serieel kan worden geordend (of waarvan dit ongedaan kan worden gemaakt) met behulp van AMF3. (Dit omvat de meeste klassen, inclusief aangepaste klassen, maar sluit bepaalde andere klassen uit, waaronder weergaveobjecten en objecten die weergaveobjecten als eigenschappen hebben.)

String komt overeen met de klasse String (equivalent aan de kolomaffiniteit TEXT).

XML komt overeen met de ActionScript (E4X) klasse XML.

XMLList komt overeen met de ActionScript (E4X) klasse XMLList.

De volgende letterlijke waarden worden standaard niet ondersteund in SQLite, maar worden wel ondersteund in Adobe AIR:

true gebruikt voor het weergeven van de letterlijke booleaanse waarde true, om met BOOLEAN-kolommen te werken.

false gebruikt voor het weergeven van de letterlijke booleaanse waarde false, om met BOOLEAN-kolommen te werken.

Ondersteuning van gegevenstypen

In tegenstelling tot de meeste SQL-databases vereist de SQL-database-engine van Adobe AIR niet dat tabelkolommen waarden van een bepaald type bevatten. Dit wordt ook niet afgedwongen. In plaats daarvan gebruikt de runtime twee concepten, opslagklassen en kolomaffiniteit, voor het beheer van gegevenstypen. In dit gedeelte worden opslagklassen en kolomaffiniteit beschreven, alsmede de manier waarop verschillen in gegevenstype onder verschillende omstandigheden worden opgelost:

- "Opslagklassen" op pagina 1200
- "Kolomaffiniteit" op pagina 1200

- "Gegevenstypen en vergelijkingsoperatoren" op pagina 1203
- "Gegevenstypen en wiskundige operatoren" op pagina 1204
- "Gegevenstypen en sorteren" op pagina 1204
- "Gegevenstypen en groeperen" op pagina 1204
- "Gegevenstypen en samengestelde SELECT-instructies" op pagina 1204

Opslagklassen

Opslagklassen vertegenwoordigen de feitelijke gegevenstypen die worden gebruikt voor het opslaan van waarden in een database. De volgende opslagklassen worden gebruikt door de database:

NULL De waarde bestaat uit een NULL-waarde.

INTEGER De waarde bestaat uit een geheel getal met voorteken.

REAL De waarde bestaat uit een zwevende-kommagetalwaarde.

TEXT De waarde bestaat uit een teksttekenreeks (beperkt tot 256 MB).

BLOB De waarde bestaat uit een BLOB (Binary Large Object), met andere woorden: ruwe binaire gegevens (beperkt tot 256 MB).

Alle waarden die als in een SQL-instructie ingesloten letterlijke waarden worden doorgegeven aan de database, of waarden die met parameters zijn gebonden aan een geprepareerde SQL-instructie, krijgen een opslagklasse toegewezen voordat de SQL-instructie wordt uitgevoerd.

Letterlijke waarden die deel uitmaken van een SQL-instructie, krijgen de opslagklasse TEXT toegewezen als deze tussen enkele of dubbele aanhalingstekens staan, INTEGER als de letterlijke waarde wordt opgegeven als getal zonder aanhalingstekens zonder decimaalteken of exponent, REAL als de letterlijke waarde bestaat uit een getal zonder aanhalingstekens met decimaalteken of exponent, en NULL als de waarde NULL is. Letterlijke waarden met de opslagklasse BLOB worden opgegeven met behulp van de notatie X'ABCD'. Zie Letterlijke waarden in expressies voor meer informatie.

Waarden die worden opgegeven als parameters met behulp van de associatieve array SQLStatement.parameters, krijgen de opslagklasse toegewezen die de oorspronkelijke gegevenstypebinding het meest benadert. Zo worden int-waarden gebonden als INTEGER-opslagklasse, krijgen getalwaarden de opslagklasse REAL, tekenreekswaarden opslagklasse TEXT en ByteArray-objecten opslagklasse BLOB.

Kolomaffiniteit

De *affiniteit*van een kolom is het aanbevolen type voor in die kolom opgeslagen gegevens. Wanneer een waarde wordt opgeslagen in een kolom (via een instructie van het type INSERT of UPDATE), probeert de runtime die waarde om te zetten van het gegevenstype naar de opgegeven affiniteit. Als bijvoorbeeld een datumwaarde (een ActionScript- of JavaScript Date-instantie) wordt ingevoegd in een kolom waarvan de affiniteit TEXT is, wordt de datumwaarde omgezet in de tekenreeksweergave (equivalent aan het aanroepen van de toString()-methode van het object) voordat deze wordt opgeslagen in de database. Als de waarde niet kan worden omgezet in de opgegeven affiniteit, treedt er een fout op en wordt de bewerking niet uitgevoerd. Wanneer een waarde wordt opgehaald uit de database via een SELECT-instructie, wordt deze geretourneerd als instantie van de klasse die overeenkomt met de affiniteit, ongeacht of de waarde is omgezet vanuit een ander gegevenstype toen deze is opgeslagen.

Als een kolom NULL-waarden accepteert, kan de ActionScript- of JavaScript-waarde null worden gebruikt als parameterwaarde om NULL op te slaan in de kolom. Wanneer een NULL-opslagklassewaarde wordt opgehaald in een SELECT-instructie, wordt deze altijd geretourneerd als de ActionScript- of JavaScript-waarde null, ongeacht de affiniteit van de kolom. Als een kolom NULL-waarden accepteert, controleert u altijd waarden die zijn opgehaald uit die kolom, om te bepalen of deze null zijn voordat u probeert de waarden om te zetten in een niet-null-type (zoals Number of Boolean).

Elke kolom in de database krijgt een van de volgende typen affiniteiten toegewezen:

- TEXT (of String)
- NUMERIC
- INTEGER (of int)
- REAL (of Number)
- Boolean
- Date
- XML
- XMLLIST
- Object
- NONE

TEXT (of String)

In een kolom met de affiniteit TEXT of String worden alle gegevens opgeslagen met de opslagklassen NULL, TEXT of BLOB. Als er numerieke gegevens worden ingevoegd in een kolom met de affiniteit TEXT, worden deze omgezet in een tekst voordat ze worden opgeslagen.

NUMERIC

Een kolom met de affiniteit NUMERIC bevat waarden met de opslagklassen NULL, REAL of INTEGER. Wanneer tekstgegevens worden ingevoegd in een kolom met affiniteit NUMERIC, wordt gepoogd deze om te zetten in een geheel getal of echt getal voordat deze worden opgeslagen. Als de conversie slaagt, wordt de waarde opgeslagen met de opslagklasse INTEGER of REAL (zo wordt een waarde '10,05' omgezet in de opslagklasse REAL voordat deze wordt opgeslagen). Als de conversie niet kan worden uitgevoerd, treedt er een fout op. Er wordt geen poging gedaan om de waarden om te zetten in een NULL-waarde. Een waarde die wordt opgehaald uit een kolom met affiniteit NUMERIC, wordt geretourneerd als instantie van het meest specifieke numerieke type waarin de waarde kan worden ingepast. Met ander woorden, als de waarde is een positief geheel getal of 0 is, wordt deze geretourneerd als een uint-instantie. Als het een negatief geheel getal betreft, wordt deze geretourneerd als een int-instantie. Als de waarde een zwevende-kommacomponent heeft en dus geen geheel getal is, wordt deze geretourneerd als een Number-instantie.

INTEGER (of int)

Een kolom die de affiniteit INTEGER gebruikt, gedraagt zich op dezelfde manier als een kolom met de affiniteit NUMERIC, met één uitzondering. Als de waarde die moet worden opgeslagen, een REAL-waarde heeft (bijvoorbeeld een Number-instantie) zonder zwevende-kommacomponent, of als de waarde een tekstwaarde is die kan worden omgezet in een REAL-waarde zonder zwevende-kommacomponent, wordt deze omgezet in een geheel getal en opgeslagen met de opslagklasse INTEGER. Als wordt gepoogd om een REAL-waarde op te slaan met een zwevende-kommacomponent, treedt er een fout op.

REAL (of Number)

Een kolom met de affiniteit REAL of NUMBER gedraagt zich als een kolom met de affiniteit NUMERIC, met dit verschil dat geheel-getalwaarden gedwongen worden omgezet in een zwevende-kommaweergave. Een waarde in een kolom REAL wordt altijd geretourneerd uit de database als een Number-instantie.

Booleaans

In een kolom met de affiniteit Boolean worden de waarden opgeslagen als waar of onwaar. Een Boolean-kolom accepteert een waarde die voorkomt uit een ActionScript- of JavaScript Boolean-instantie. Als code probeert om een tekenreekswaarde op te slaan, wordt een tekenreeks met een lengte groter dan nul beschouwd als waar en een lege tekenreeks is onwaar. Als code probeert om numerieke gegevens op te slaan, wordt elke niet-nulwaarde opgeslagen als waar en wordt 0 opgeslagen als onwaar. Wanneer een booleaanse waarde wordt opgehaald via een SELECT-instructie, wordt deze geretourneerd als een Boolean-instantie. Niet-NULL-waarden worden opgeslagen met de opslagklasse INTEGER (0 voor onwaar en 1 voor waar) en worden omgezet in booleaanse objecten wanneer gegevens worden opgehaald.

Date

In een kolom met de affiniteit Date worden datum- en tijdwaarden opgeslagen. Een Date-kolom is bedoeld voor het accepteren van waarden die voortkomen uit ActionScript- of JavaScript Date-instanties. Als wordt geprobeerd een tekenreekswaarde op te slaan in een kolom met affiniteit Date, probeert de runtime deze waarde om te zetten in een Juliaanse datum. Als de conversie mislukt, treedt er een fout op. Als code probeert om een Number-, int- of uint-waarde op te slaan, wordt niet gepoogd om de gegevens te valideren en wordt aangenomen dat het een geldige Juliaanse-datumwaarde betreft. Een Date-waarde die wordt opgehaald via een SELECT-instructie, wordt automatisch omgezet in een Date-instantie. Date-waarden worden opgeslagen als Juliaanse-datumwaarden met de opslagklasse REAL, waardoor sorteer- en vergelijkingsbewerkingen naar verwachting werken.

XML of XMLList

In een kolom met de affiniteit XML of XMLList, worden XML-structuren opgeslagen. Wanneer code probeert om gegevens op te slaan in een XML-kolom met een SQLStatement-parameter, probeert de runtime de waarde om te zetten en te valideren met behulp van de XML()- of XMLList()-functie van ActionScript. Als de waarde niet kan worden omgezet in geldige XML, treedt er een fout op. Als bij de poging tot opslag van de gegevens een letterlijke SQL-tekstwaarde wordt gebruikt (bijvoorbeeld INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)')), wordt de waarde niet geparseerd of gevalideerd. Hiervan wordt aangenomen dat deze goed gestructureerd is. Wanneer een ongeldige waarde wordt opgeslagen, wordt deze bij ophalen geretourneerd als leeg XML-object. XML- en XMLList-gegevens worden opgeslagen met de opslagklasse TEXT of NULL.

Object

In een kolom met de affiniteit Object worden complexe ActionScript- of JavaScript-objecten opgeslagen, inclusief instanties van objectklassen alsook instanties van objectsubklassen zoals Array-instanties en zelfs instanties van aangepaste klassen. Object-kolomgegevens worden serieel geordend in de AMF3-indeling en opgeslagen met de opslagklasse BLOB. Wanneer een waarde wordt opgehaald, wordt de seriële ordening vanuit AMF3 ongedaan gemaakt en wordt de waarde geretourneerd als instantie van de klasse zoals deze werd opgeslagen. Houd er rekening mee dat de seriële ordening van bepaalde ActionScript-klassen, met name weergaveobjecten, niet ongedaan kan worden gemaakt om terug te gaan naar instanties van het oorspronkelijke gegevenstype. Voordat een instantie van een aangepaste klasse kan worden opgeslagen, moet u een alias voor de klasse registreren met de methode flash.net.registerClassAlias() (of in Flex door [RemoteObject]-metagegevens toe te voegen aan de klassendeclaratie). Voordat u de gegevens ophaalt, moet u dezelfde alias ook registreren voor de klasse. Alle gegevens waarvan de seriële ordening niet op de juiste manier ongedaan kan worden gemaakt, hetzij omdat de seriële ordening van de klasse van nature niet ongedaan kan worden gemaakt, hetzij omdat er een ontbrekende of niet-overeenstemmende klassenalias is, worden geretourneerd als anoniem object (een exemplaar van de klasse Object) met eigenschappen en waarden die overeenkomen met de oorspronkelijke instantie zoals deze is opgeslagen.

NONE

Een kolom met de affiniteit NONE heeft geen voorkeur een bepaalde opslagklasse. Er wordt geen poging ondernomen om de gegevens om te zetten voordat deze worden ingevoegd.

Affiniteit bepalen

Het type affiniteit van een kolom wordt bepaald door het gedeclareerde type van de kolom in de instructie CREATE TABLE. Bij het bepalen van het type worden de volgende regels (niet-hoofdlettergevoelig) toegepast:

- Als het gegevenstype van de kolom een van de tekenreeksen CHAR, CLOB, STRI of TEXT bevat, heeft die kolom de affiniteit TEXT/String. U ziet dat het type VARCHAR de tekenreeks CHAR bevat en daarom de affiniteit TEXT krijgt toegewezen.
- Als het gegevenstype voor de kolom de tekenreeks BLOB bevat of als er geen gegevenstype is opgegeven, heeft de kolom de affiniteit NONE.
- Als het gegevenstype voor de kolom de tekenreeks XMLL bevat, heeft de kolom de affiniteit XMLList.
- Als het gegevenstype voor de kolom de tekenreeks XML bevat, heeft de kolom de affiniteit XML.
- Als het gegevenstype voor de kolom de tekenreeks OBJE bevat, heeft de kolom de affiniteit Object.
- Als het gegevenstype voor de kolom de tekenreeks BOOL bevat, heeft de kolom de affiniteit Boolean.
- Als het gegevenstype voor de kolom de tekenreeks DATE bevat, heeft de kolom de affiniteit Date.
- Als het gegevenstype voor de kolom de tekenreeks INT (inclusief UINT) bevat, is de toegewezen affiniteit INTEGER/int.
- Als het gegevenstype voor de kolom een van de tekenreeksen REAL, NUMB, FLOA of DOUB bevat, heeft de kolom de affiniteit REAL/Number.
- In alle andere gevallen is de affiniteit NUMERIC.
- Als een tabel wordt gemaakt met een instructie CREATE TABLE t AS SELECT..., wordt voor geen van de kolommen een gegevenstype opgegeven en krijgen deze de affiniteit NONE toegewezen.

Gegevenstypen en vergelijkingsoperatoren

De volgende binaire vergelijkingsoperatoren =, <, <=, >= en != worden ondersteund, samen met een bewerking voor het testen van het lidmaatschap, IN, en de ternaire vergelijkingsoperator BETWEEN. Zie Operatoren voor details over deze operatoren.

De resultaten van een vergelijking zijn afhankelijk van de opslagklassen van de twee waarden die worden vergeleken. Bij de vergelijking van twee waarden worden de volgende regels toegepast:

- Een waarde met opslagklasse NULL wordt beschouwd als kleiner dan elke andere waarde (inclusief een andere waarde met de opslagklasse NULL).
- Een INTEGER- of REAL-waarde is kleiner dan een willekeurige TEXT- of BLOB-waarde. Wanneer een INTEGER- of REAL-waarde wordt vergeleken met een andere INTEGER- of REAL-waarde, wordt een numerieke vergelijking uitgevoerd.
- Een TEXT-waarde is kleiner dan een BLOB-waarde. Bij de vergelijking van twee TEXT-waarden wordt een binaire vergelijking uitgevoerd.
- Wanneer twee BLOB-waarden worden vergeleken, wordt het resultaat altijd bepaald met een binaire vergelijking.

De ternaire operator BETWEEN wordt altijd omgevormd tot de equivalente binaire expressie. Zo wordt a BETWEEN b AND c omgevormd tot a >= b AND a <= c, zelfs als dit inhoudt dat er andere affiniteiten worden toegepast op a in elk van de vergelijkingen die nodig zijn voor het evalueren van de expressie.

Expressies van het type a IN (SELECT b ....) worden afgehandeld met de drie regels die eerder zijn opgenoemd voor binaire vergelijkingen, namelijk op een vergelijkbare manier als a = b. Als bijvoorbeeld b een kolomwaarde is en a een expressie, wordt de affiniteit van b toegepast op a voor alle vergelijkingen die plaatsvinden. De expressie a IN (x, y, z) wordt omgevormd tot a = +x OR a = +y OR a = +z. De waarden rechts van de operator IN (de waarden x, y en z in dit voorbeeld) worden als expressies beschouwd, zelfs als deze toevallig kolomwaarden zijn. Als de waarde links van de operator IN een kolom is, wordt de affiniteit van die kolom gebruikt. Als de waarde een expressie is, vindt geen conversie plaats.

Hoe vergelijkingen worden uitgevoerd, kan ook worden beďnvloed door het gebruik van een COLLATE-component. Zie COLLATE voor meer informatie.

Gegevenstypen en wiskundige operatoren

Voor elk van de ondersteunde wiskundige operatoren, *, /, %, + en -, wordt de affiniteit NUMERIC toegepast op elke operand voordat de expressie wordt geëvalueerd. Als een bepaalde operand niet kan worden omgezet in de opslagklasse NUMERIC, leidt de evaluatie van de expressie tot NULL.

Wanneer de samenvoegingsoperator || wordt gebruikt, wordt elke operand omgezet naar de opslagklasse TEXT voordat de expressie wordt geëvalueerd. Als een bepaalde operand niet kan worden omgezet in de opslagklasse TEXT, is het resultaat van de expressie NULL. In twee situaties kunnen operanden mogelijk niet worden omgezet: als de waarde van de operand NULL is, of als het een BLOB betreft met andere opslagklasse dan TEXT.

Gegevenstypen en sorteren

Wanneer waarden worden gesorteerd met een ORDER BY-component, komen waarden met opslagklasse NULL op de eerste plaats. Deze worden gevolgd door INTEGER- en REAL-waarden die elkaar afwisselen in een numerieke volgorde, gevolgd door TEXT-waarden in binaire volgorde of op basis van de opgegeven sortering (BINARY of NOCASE). Als laatste komen de waarden BLOB in binaire volgorde. Vóór de sortering vinden er geen omzettingen van opslagklassen plaats.

Gegevenstypen en groeperen

Bij het groeperen van waarden met de GROUP BY-component worden waarden met verschillende opslagklassen als afzonderlijk van elkaar beschouwd. Een uitzondering hierop vormen echter INTEGER- en REAL-waarden die als gelijkwaardig aan elkaar worden beschouwd als deze numeriek equivalent zijn. Als resultaat van een GROUP BY-component worden op geen van de waarden affiniteiten toegepast.

Gegevenstypen en samengestelde SELECT-instructies

De samengestelde SELECT-operatoren UNION, INTERSECT en EXCEPT voeren impliciete vergelijkingen van waarden uit. Voordat deze vergelijkingen worden uitgevoerd, wordt op elke waarde mogelijk een affiniteit toegepast. Indien er al een affiniteit is, wordt dezelfde affiniteit toegepast op alle waarden die kunnen worden geretourneerd in één kolom van de samengestelde SELECT-resultatenset. De affiniteit die wordt toegepast, is de affiniteit van de kolom die wordt geretourneerd door de eerste samengestelde SELECT-instructie die op die positie een kolomwaarde heeft (en niet een of andere expressie). Als voor een bepaalde samengestelde SELECT-kolom geen van de samengestelde SELECT-instructies een kolomwaarde retourneert, wordt er geen affiniteit toegepast op de waarden van die kolom voordat er wordt vergeleken.

Hoofdstuk 67: Foutberichten, id's en argumenten voor SQL

De klasse SQLError vertegenwoordigt diverse fouten die kunnen optreden bij het werken met een lokale SQL-database van Adobe AIR. Voor elke gegeven uitzondering heeft de SQLError-instantie een eigenschap `details` met een Engelstalig foutbericht. Daarnaast heeft elk foutbericht een bijbehorende unieke id die beschikbaar is in de eigenschap `detailID` van het SQLError-object Aan de hand van de eigenschap `detailID` kan een toepassing het specifieke gedetailleerde foutbericht herkennen. De toepassing kan dan een alternatieve tekst leveren voor de eindgebruiker in de taal die hoort bij de geselecteerde landinstelling. De argumentwaarden in de array `detailArguments` kunnen dienen als vervanging voor de desbetreffende positie in de tekenreeks van het foutbericht. Dit is handig voor toepassingen die het foutbericht van de eigenschap `details` voor een fout rechtstreeks weergeven voor eindgebruikers met een specifieke landinstelling.

De volgende tabel bevat een lijst met de waarden van `detailID` en de bijbehorende Engelstalige tekst van het foutbericht. Variabelen in de berichten geven aan waar de waarden van `detailArguments` terecht komen in runtime. Deze lijst kan worden gebruikt als bron voor het vertalen van de foutberichten die kunnen optreden bij SQL-databasebewerkingen.

SQLError-detailID	Engelstalig foutbericht en parameters
1001	Connection closed (Verbinding gesloten).
1102	Database must be open to perform this operation (Database moet zijn geopend om deze bewerking uit te voeren).
1003	%s [,|and %s] parameter name(s) found in parameters property but not in the SQL specified (parameternaam/-namen %s [,|en %s] wel gevonden in parametereigenschap, maar niet in de opgegeven SQL).
1004	Mismatch in parameter count (Ongelijk aantal parameters). Found %d in SQL specified and %d value(s) set in parameters property (%d gevonden in opgegeven SQL en waarde(n) %d ingesteld in parametereigenschap). Expecting values for %s [,|and %s] (Waarden voor %s [,|en %s] verwacht).
1005	Auto compact could not be turned on (Kan automatisch comprimeren niet inschakelen).
1006	The pageSize value could not be set (Kan de waarde pageSize niet instellen).
1007	The schema object with name '%s' of type '%s' in database '%s' was not found (Het schemaobject met de naam '%s' van het type '%s' in de database '%s' is niet gevonden).
1008	The schema object with name '%s' in database '%s' was not found (Het schemaobject met de naam '%s' in de database '%s' is niet gevonden).
1009	No schema objects with type '%s' in database '%' were found (Geen schemaobjecten van het type '%s' gevonden in de database '%').
1010	No schema objects in database '%s' were found (Geen schemaobjecten gevonden in de database '%s').
2001	Parser stack overflow. (Parseerstapeloverloop.)
2002	Too many arguments on function '%s'. (Te veel argumenten voor functie '%s'.)
2003	near '%s': syntax error. (bij '%s': syntaxisfout.)
2004	there is already another table or index with this name: '%s'. (er is al een andere tabel of index met deze naam: '%s'.)
2005	PRAGMA is not allowed in SQL. (PRAGMA is niet toegestaan in SQL.)
2006	Not a writable directory. (Geen schrijfbare directory.)
2007	Unknown or unsupported join type: '%s %s %s'. (Onbekend of niet-ondersteund join-type: '%s %s %s'.)
2008	RIGHT and FULL OUTER JOINs are not currently supported. (RIGHT- en FULL OUTER-JOINs worden momenteel niet ondersteund.)
2009	A NATURAL join may not have an ON or USING clause. (Een NATURAL-join mag geen ON- of USING-component bevatten.)

2010	Cannot have both ON and USING clauses in the same join. (Kan geen ON- en USING-component hebben in dezelfde join.)
2011	Cannot join using column '%s' - column not present in both tables. (Geen join mogelijk op kolom '%s' - kolom niet aanwezig in beide tabellen.)
2012	Only a single result allowed for a SELECT that is part of an expression. (Alleen enkelvoudig resultaat toegestaan voor een SELECT die onderdeel vormt van een expressie.)
2013	No such table: '[%s.]%s'. (Geen tabel: '[%s.]%s'.)
2014	No tables specified. (Geen tabellen opgegeven.)
2015	Too many columns in result set|too many columns on '%s'. (Te veel kolommen ingesteld in resultatenset|te veel kolommen in '%s'.)
2016	%s ORDER|GROUP BY term out of range - should be between 1 and %d (%s ORDER|GROUP BY-term buiten bereik - moet liggen tussen 1 en %d)
2017	Too many terms in ORDER BY clause. (Te veel termen in ORDER BY-component.)
2018	%s ORDER BY term out of range - should be between 1 and %d. (%s ORDER BY-term buiten bereik - moet liggen tussen 1 en %d.)
2019	%r ORDER BY term does not match any column in the result set. (%r ORDER BY-term komt niet overeen met een kolom in resultatenset.)
2020	ORDER BY clause should come after'%s' not before. (ORDER BY-component moet na '%s' staan en niet ervoor.)
2021	LIMIT clause should come after'%s' not before. (LIMIT-component moet na '%s' staan en niet ervoor.)
2022	SELECTs to the left and right of '%s' do not have the same number of result columns. (SELECT-exemplaren links en rechts van '%s' hebben niet hetzelfde aantal resultaatkolommen.)
2023	A GROUP BY clause is required before HAVING. (Een GROUP BY-component is vereist vóór HAVING.)
2024	Aggregate functions are not allowed in the GROUP BY clause. (Statistische functies zijn niet toegestaan in de GROUP BY-component.)
2025	DISTINCT in aggregate must be followed by an expression. (DISTINCT in statistische functie moet worden gevolgd door een expressie.)
2026	Too many terms in compound SELECT. (Te veel termen in samengestelde SELECT.)
2027	Too many terms in ORDER|GROUP BY clause. (Te veel termen in ORDER-|GROUP BY-component.)
2028	Temporary trigger may not have qualified name. (Tijdelijke trigger mag geen gekwalificeerde naam hebben.)
2030	Trigger '%s' already exists. (Trigger '%s' bestaat al.)
2032	Cannot create BEFORE|AFTER trigger on view: '%s'. (Kan geen trigger BEFORE|AFTER maken voor weergave:'%s'.)
2033	Cannot create INSTEAD OF trigger on table: '%s'. (Kan geen trigger INSTEAD OF maken voor tabel: '%s'.)
2034	No such trigger: '%s'. (Geen trigger: '%s'.)
2035	Recursive triggers not supported ('%s'). (Recursieve triggers niet ondersteund ('%s'.)
2036	No such column: %s[. (Geen kolom: %s[.)%s[.%s]]
2037	VACUUM is not allowed from SQL. (VACUUM is niet toegestaan door SQL)
2043	Table '%s': indexing function returned an invalid plan. (Tabel '%s': indexeringsfunctie heeft een ongeldig plan geretourneerd.)
2044	At most %d tables in a join. (Maximaal %d tabellen in een join.)
2046	Cannot add a PRIMARY KEY column. (Kan geen kolom PRIMARY KEY toevoegen.)
2047	Cannot add a UNIQUE column. (Kan geen kolom UNIQUE toevoegen.)
2048	Cannot add a NOT NULL column with default value NULL. (Kan geen kolom NOT NULL toevoegen met standaardwaarde NULL.)
2049	Cannot add a column with non-constant default. (Kan geen kolom toevoegen met niet-constante standaardwaarde.)
2050	Cannot add a column to a view. (Kan geen kolom toevoegen aan een weergave.)
2051	ANALYZE is not allowed in SQL. (ANALYZE is niet toegestaan in SQL.)
2052	Invalid name: '%s'. (Ongeldige naam: '%s'.)
2053	ATTACH is not allowed from SQL. (ATTACH is niet toegestaan door SQL)
2054	%s '%s' cannot reference objects in database '%s' (%s '%s' kan niet verwijzen naar objecten in database '%s')
2055	Access to '[%s.]%s.%s' is prohibited. (Toegang tot '[%s.]%s.%s' is verboden.)

2056	Not authorized. (Niet geautoriseerd.)
2058	No such view: '[%s.]%s'. (Geen weergave: '[%s.]%s'.)
2060	Temporary table name must be unqualified. (Tijdelijke tabelnaam moet ongekwalificeerd zijn.)
2061	Table '%s' already exists. (Tabel '%s' bestaat al.)
2062	There is already an index named: '%s'. (Er is al een index met de naam: '%s'.)
2064	Duplicate column name: '%s'. (Dubbele kolomnaam: '%s'.)
2065	Table '%s' has more than one primary key. (Tabel '%s' heeft meer dan één primaire sleutel.)
2066	AUTOINCREMENT is only allowed on an INTEGER PRIMARY KEY (AUTOINCREMENT is alleen toegestaan voor een INTEGER PRIMARY KEY)
2067	No such collation sequence: '%s'. (Geen sorteervolgorde: '%s'.)
2068	Parameters are not allowed in views. (Parameters zijn niet toegestaan in weergaven.)
2069	View '%s' is circularly defined. (Weergave '%s' is circulair gedefinieerd.)
2070	Table '%s' may not be dropped. (Tabel '%s' mag niet worden verwijderd.)
2071	Use DROP VIEW to delete view '%s'. (Gebruik DROP VIEW voor het verwijderen van weergave '%s'.)
2072	Use DROP TABLE to delete table '%s'. (Gebruik DROP TABLE voor het verwijderen van tabel '%s'.)
2073	Foreign key on '%s' should reference only one column of table. (Externe sleutel voor '%s' moet verwijzen naar slechts één kolom van tabel '%s'.)
2074	Number of columns in foreign key does not match the number of columns in the referenced table. (Aantal kolommen in externe sleutel komt niet overeen met het aantal kolommen in de tabel waarnaar wordt verwezen.)
2075	Unknown column '%s' in foreign key definition. (Onbekende kolom '%s' in definitie van externe sleutel.)
2076	Table '%s' may not be indexed. (Tabel '%s' mag niet worden geïndexeerd.)
2077	Views may not be indexed. (Weergaven mogen niet worden geïndexeerd.)
2080	Conflicting ON CONFLICT clauses specified. (Conflicterende ON CONFLICT-componenten opgegeven.)
2081	No such index: '%s'. (Geen index: '%s'.)
2082	Index associated with UNIQUE or PRIMARY KEY constraint cannot be dropped. (Index die is gekoppeld aan de beperking UNIQUE of PRIMARY KEY, kan niet worden verwijderd.)
2083	BEGIN is not allowed in SQL. (BEGIN is niet toegestaan in SQL.)
2084	COMMIT is not allowed in SQL. (COMMIT is niet toegestaan in SQL.)
2085	ROLLBACK is not allowed in SQL. (ROLLBACK is niet toegestaan in SQL.)
2086	Unable to open a temporary database file for storing temporary tables. (Kan tijdelijk databasebestand voor de opslag van tijdelijke tabellen niet openen.)
2087	Unable to identify the object to be reindexed. (Kan opnieuw te indexeren object niet identificeren.)
2088	Table '%s' may not be modified. (Tabel '%s' mag niet worden gewijzigd.)
2089	Cannot modify '%s' because it is a view. (Kan '%s' niet wijzigen omdat het een weergave is.)
2090	Variable number must be between ?0 and ?%d<. (Variabel getal moet liggen tussen ?0 en ?%d<.)
2092	Misuse of aliased aggregate '%s'. (Verkeerd gebruik van statistische functie '%s' met alias.)
2093	Ambiguous column name: '[%s.[%s.]]%s'. (Ambigue kolomnaam: '[%s.[%s.]]%s'.)
2094	No such function: '%s'. (Geen functie: '%s'.)
2095	Wrong number of arguments to function '%s'. (Onjuist aantal argumenten voor de functie '%s'.)
2096	Subqueries prohibited in CHECK constraints. (Subquery's verboden in CHECK-beperkingen.)
2097	Parameters prohibited in CHECK constraints. (Parameters verboden in CHECK-beperkingen.)
2098	Expression tree is too large (maximum depth %d) (Expressiestructuur is te groot (maximale diepte %d))
2099	RAISE() may only be used within a trigger-program (RAISE() mag alleen worden gebruikt binnen een triggerprogramma)

2100	Table '%s' has %d columns but %d values were supplied. (Tabel '%s' heeft %d kolommen, maar er zijn %d waarden opgegeven.)
2101	Database schema is locked: '%s'. (Databaseschema is vergrendeld: '%s'.)
2102	Statement too long. (Instructie te lang.)
2103	Unable to delete/modify collation sequence due to active statements (Kan sorteervolgorde niet verwijderen/wijzigen vanwege actieve instructies)
2104	Too many attached databases - max %d. (Te veel gekoppelde databases - max. %d.)
2105	Cannot ATTACH database within transaction. (ATTACH van database binnen transactie niet mogelijk.)
2106	Database '%s' is already in use. (Database '%s' wordt al gebruikt.)
2108	Attached databases must use the same text encoding as main database. (Gekoppelde databases moeten dezelfde tekstcodering gebruiken als de hoofddatabase.)
2200	Onvoldoende geheugen.
2201	Unable to open database. (Kan database niet openen.)
2202	Cannot DETACH database within transaction. (DETACH van database binnen transactie niet mogelijk.)
2203	Cannot detach database: '%s'. (Kan database niet ontkoppelen: '%s'.)
2204	Database '%s' is locked. (Database '%s' is vergrendeld.)
2205	Unable to acquire a read lock on the database. (Kan geen leesbeveiliging op de database aanbrengen.)
2206	[column|columns] '%s'[,'%s'] are not [unique|is] not unique. ([kolom|kolommen] '%s'[,'%s'] zijn niet [uniek|is] niet uniek.)
2207	Malformed database schema. (Databaseschema met ongeldige indeling.)
2208	Unsupported file format. (Niet-ondersteunde bestandsindeling.)
2209	Unrecognized token: '%s'. (Niet-herkend token: '%s'.)
2300	Could not convert text value to numeric value. (Kan tekstwaarde niet omzetten naar numerieke waarde.)
2301	Could not convert string value to date. (Kan tekenreekswaarde niet omzetten naar datum.)
2302	Could not convert floating point value to integer without loss of data. (Kan zwevende-kommawaarde niet omzetten naar geheel getal zonder gegevensverlies.)
2303	Cannot rollback transaction - SQL statements in progress. (Kan transactie niet terugdraaien - SQL-instructies worden uitgevoerd.)
2304	Cannot commit transaction - SQL statements in progress. (Kan transactie niet vastleggen - SQL-instructies worden uitgevoerd.)
2305	Database table is locked: '%s'. (Databasetabel is vergrendeld: '%s'.)
2306	Read-only table. (Alleen-lezen tabel.)
2307	String or blob too big. (Tekenreeks of blob te groot.)
2309	Cannot open indexed column for writing. (Kan geďndexeerde kolom niet openen voor schrijven.)
2400	Cannot open value of type %s. (Kan waarde van type %s niet openen.)
2401	No such rowid: %s<. (Geen rij-id: %s<.)
2402	Object name reserved for internal use: '%s'. (Objectnaam gereserveerd voor intern gebruik: '%s')
2403	View '%s' may not be altered. (Weergave '%s' mag niet worden gewijzigd.)
2404	Default value of column '%s' is not constant. (Standaardwaarde van kolom '%s' is niet constant.)
2405	Not authorized to use function '%s'. (Niet bevoegd tot gebruik van functie '%s'.)
2406	Misuse of aggregate function '%s'. (Verkeerd gebruik van statistische functie '%s'.)
2407	Misuse of aggregate: '%s'. (Verkeerd gebruik van statistische functie: '%s'.)
2408	No such database: '%s'. (Geen database: '%s'.)
2409	Table '%s' has no column named '%s'. (Tabel '%s' heeft geen kolom met de naam '%s'.)
2501	No such module: '%s'. (Geen module: '%s'.)
2508	No such savepoint: '%s'. (Geen opslagpunt: '%s'.)
2510	Cannot rollback - no transaction is active. (Rollback niet mogelijk: geen transactie is actief.)
2511	Cannot commit - no transaction is active. (Kan niet vastleggen: geen transactie is actief.)

Hoofdstuk 68: AGAL (Adobe Graphics Assembly Language)

AGAL (Adobe Graphics Assembly Language) is een shadertaal voor het definiëren van hoekpunt- en fragmentrenderprogramma's. De AGAL-programma's dienen in de binaire bytecode-indeling die in dit document wordt beschreven te worden geüpload naar de renderingcontext.

AGAL-bytecode-indeling

AGAL-bytecode moet de Endian.LITTLE_ENDIAN-indeling gebruiken.

Bytecodeheader

AGAL-bytecode moet beginnen met een uit zeven bytes bestaande header:

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

Offset (bytes)	Grootte (bytes)	Naam	Beschrijving
0	1	magic	moet 0xa0 zijn
1	4	versie	moet 1 zijn
5	1	shadertype-id	moet 0xa1 zijn
6	1	shadertype	0 voor een hoekpuntprogramma; 1 voor een fragmentprogramma

Tokens

De header wordt onmiddellijk gevolgd door een aantal tokens. Elk token is 192 bits (24 bytes) groot en heeft altijd de volgende indeling:

```
[opcode][destination][source1][source2 of sampler]
```

Niet elke op-code gebruikt al deze velden. Niet-gebruikte velden moet worden ingesteld op 0.

Bewerkingscodes

Het [opcode]-veld is 32 bits groot en kan een van de volgende waarden hebben:

Naam	Op-code	Bewerking	Beschrijving
mov	0x00	move	gegevens verplaatsen van bron1 naar doel, componentsgewijs
add	0x01	add	doel = bron1 + bron2, componentsgewijs
sub	0x02	subtract	doel = bron1 - bron2, componentsgewijs
mul	0x03	meerdere	doel = bron1 * bron2, componentsgewijs
div	0x04	delen	doel = bron1 / bron2, componentsgewijs
rcp	0x05	wederkerig	doel = 1/bron1, componentsgewijs

Naam	Op-code	Bewerking	Beschrijving
min	0x06	minimum	doel = minimum(bron1,bron2), componentsgewijs
max	0x07	maximum	doel = maximum(bron1,bron2), componentsgewijs
frc	0x08	fractioneel	doel = bron1 - (zwevend)ondergrens(bron1), componentsgewijs
sqt	0x09	vierkantswortel	doel = sqrt(bron1), componentsgewijs
rsq	0x0a	wederkerige vierkantswortel	doel = 1/sqrt(bron1), componentsgewijs
pow	0x0b	macht	doel = pow(bron1,bron2), componentsgewijs
log	0x0c	logaritme	doel = log_2(bron1), componentsgewijs
exp	0x0d	exponentieel	doel = 2^bron1, componentsgewijs
nrm	0x0e	normalize	doel = normalize(bron1), componentsgewijs (produceert alleen een resultaat met drie componenten, het doel moet worden gemaskeerd in .xyz of minder)
sin	0x0f	sinus	doel = sin(bron1), componentsgewijs
cos	0x10	cosinus	doel = cos(bron1), componentsgewijs
crs	0x11	kruisproduct	doel.x = bron1.y * bron2.z - bron1.z * bron2.y doel.y = bron1.z * bron2.x - bron1.x * bron2.z doel.z = bron1.x * bron2.y - bron1.y * bron2.x (produceert alleen een resultaat met 3 componenten, het doel moet worden gemaskeerd in .xyz of minder)
dp3	0x12	scalair product	doel = bron1.x*bron2.x + bron1.y*bron2.y + bron1.z*bron2.z
dp4	0x13	scalair product	doel = bron1.x*bron2.x + bron1.y*bron2.y + bron1.z*bron2.z + bron1.w*bron2.w
abs	0x14	absoluut	doel = abs(bron1), componentsgewijs
neg	0x15	negatief maken	doel = -bron1, componentsgewijs
sat	0x16	satureren	doel = maximum(minimum(bron1,1),0), componentsgewijs
m33	0x17	matrix vermenigvuldigen 3x3	doel.x = (bron1.x * bron2[0].x) + (bron1.y * bron2[0].y) + (bron1.z * bron2[0].z) doel.y = (bron1.x * bron2[1].x) + (bron1.y * bron2[1].y) + (bron1.z * bron2[1].z) doel.z = (bron1.x * bron2[2].x) + (bron1.y * bron2[2].y) + (bron1.z * bron2[2].z) (produceert alleen een resultaat met 3 componenten, het doel moet worden gemaskeerd in .xyz of minder)
m44	0x18	matrix vermenigvuldigen 4x4	doel.x = (bron1.x * bron2[0].x) + (bron1.y * bron2[0].y) + (bron1.z * bron2[0].z) + (bron1.w * bron2[0].w) doel.y = (bron1.x * bron2[1].x) + (bron1.y * bron2[1].y) + (bron1.z * bron2[1].z) + (bron1.w * bron2[1].w) doel.z = (bron1.x * bron2[2].x) + (bron1.y * bron2[2].y) + (bron1.z * bron2[2].z) + (bron1.w * bron2[2].w) doel.w = (bron1.x * bron2[3].x) + (bron1.y * bron2[3].y) + (bron1.z * bron2[3].z) + (bron1.w * bron2[3].w)

Naam	Op-code	Bewerking	Beschrijving
m34	0x19	matrix vermenigvuldigen 3x4	doel.x = (bron1.x * bron2[0].x) + (bron1.y * bron2[0].y) + (bron1.z * bron2[0].z) + (bron1.w * bron2[0].w) doel.y = (bron1.x * bron2[1].x) + (bron1.y * bron2[1].y) + (bron1.z * bron2[1].z) + (bron1.w * bron2[1].w) doel.z = (bron1.x * bron2[2].x) + (bron1.y * bron2[2].y) + (bron1.z * bron2[2].z) + (bron1.w * bron2[2].w) (produceert alleen een resultaat met 3 componenten, het doel moet worden gemaskeerd in .xyz of minder)
kil	0x27	verwijderen/negeren (alleen fragmentshader)	Als de enige scalaire broncomponent minder is dan nul, wordt het fragment genegeerd en niet naar de framebuffer getekend. (Het doelregister moet volledig ingesteld zijn op 0).
tex	0x28	structuurvoorbeeld (alleen fragmentshader)	doel is gelijk aan laden van bron2 van structuur op bron1 van coördinaten. In dit geval moet bron2 de samplerindeling hebben.
sge	0x29	instellen als groter of gelijk aan	doel = bron1 >= bron2 ? 1 : 0, componentsgewijs
slt	0x2a	instellen als kleiner dan	doel = bron1 < bron2 ? 1 : 0, componentsgewijs
seq	0x2c	instellen indien gelijk	doel = bron1 == bron2 ? 1 : 0, componentsgewijs
sne	0x2d	instellen indien niet gelijk	doel = bron1 != bron2 ? 1 : 0, componentsgewijs

In AGAL2 zijn de volgende op-codes geïntroduceerd:

Naam	Op-code	Bewerking	Beschrijving
ddx	0x1a	gedeeltelijk derivaat in X	Gedeeltelijk derivaat in X van bron1 naar doel laden.
ddy	0x1b	gedeeltelijk derivaat in Y	Gedeeltelijk derivaat in Y van bron1 naar doel laden.
ife	0x1c	indien gelijk aan	Springen als bron1 gelijk is aan bron2.
ine	0x1d	indien niet gelijk aan	Springen als bron1 niet gelijk is aan bron2.
ifg	0x1e	indien groter dan	Springen als bron1 groter dan of gelijk is aan bron2.
ifl	0x1f	indien kleiner dan	Springen als bron1 kleiner is dan bron2.
els	0x20	else	Else-blok
eif	0x21	Endif	If-blok of else-blok sluiten.

Indeling van het veld Doel

Het veld [doel] is 32 bits groot:

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = Registertype (4 bits)

M = Schrijfmasker (4 bits)

N = Registernummer (16 bits)

- = niet-gedefinieerd, moet 0 zijn

Indeling van het veld Bron

Het veld [bron] is 64 bits groot:

```
63.............................................................0
D-------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

D = Direct=0/Indirect=1 voor direct worden Q en I genegeerd, 1 bit

Q = Indexregistercomponent selecteren (2 bits)

I = Indexregistertype (4 bits)

T = Registertype (4 bits)

S = Swizzle (8 bits, 2 bits per component)

O = Indirecte offset (8 bits)

N = Registernummer (16 bits)

- = niet-gedefinieerd, moet 0 zijn

Indeling van het veld Sampler

Het tweede bronveld voor de tex-op-code moet de [sampler]-indeling van 64 bits groot hebben:

```
63.............................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = Samplerregisternummer (16 bits)

B = Structuur-LOD-afwijking (Level-Of-Detail), geheel getal met teken, schalen factor 8. De gebruikte zwevende-kommawaarde is b/8.0 (8 bits)

T = Registertype, moet 5 zijn, Sampler (4 bits)

F = Filter (0=dichtstbijzijnd,1=lineair) (4 bits)

M = Mipmap (0=uitschakelen,1=dichtstbijzijnd, 2=lineair)

W = Insluiten (0=vastzetten,1=herhalen)

S = Speciale-markeringbits (moet 0 zijn)

D = Dimensie (0=2D, 1=Kubus)

Programmaregisters

Het aantal gebruikte registers hangt af van het Context3D-profiel dat wordt gebruikt. Het aantal registers en hun gebruiksgegevens worden gedefinieerd in de volgende tabel:

Naam **AGAL2** **AGAL3** **Gebruik**	**Waarde**	**AGAL**	
Aantal per fragmentprogramma Aantal per hoekpuntprogramma Aantal per fragmentprogramma Aantal per hoekpuntprogramma		Aantal per fragmentprogramma	Aantal per hoekpuntprogramma
Ondersteuning van context 3D-profielen Standaard Standaard uitgebreid		Beneden standaard	
SWF-versie 25 28 en hoger		Minder dan 25	
Attribuut N.v.t. 8 N.v.t. 16 Hoekpuntshaderinvoer; lezen uit een hoekpuntbuffer die is opgegeven met Context3D.setVertexBufferAt().	0	N.v.t.	8

Naam **AGAL2** **AGAL3** **Gebruik**	**Waarde**	**AGAL**	
Constante 64 250 200 250 Shaderinvoer; ingesteld met de Context3D.setProgramConstants()-functies.	1	28	128
Tijdelijke opslag 26 26 26 26 Tijdelijk register voor berekening, niet toegankelijk buiten programma.	2	8	8

Naam **AGAL2** **AGAL3** **Gebruik**	**Waarde**	**AGAL**	
Uitvoer 1 1 1 1 Shaderuitvoer: in een hoekpuntprogramma is de uitvoer de positie van de clipruimte. In een fragmentprogramma heeft de uitvoer de vorm van een kleur.	3	1	1

Naam **AGAL2** **AGAL3** **Gebruik**	**Waarde**	**AGAL**	
Variabel 10 10 10 10 Breng geïnterpoleerde gegevens over tussen hoekpunt- en fragmentshaders. De variabele-registers uit het hoekpuntprogramma worden als invoer toegepast op het fragmentprogramma. Waarden worden geïnterpoleerd op basis van de afstand tot de hoekpunten van de driehoek.	4	8	8

Naam **AGAL2** **AGAL3** **Gebruik**	**Waarde**	**AGAL**	
Sampler 16 N.v.t. 16 N.v.t. Fragments haderinvoer; lezen vanuit een structuur die is opgegeven met Context3D.setTextureAt().	5	8	N.v.t.
Fragmentregister 1 N.v.t. 1 N.v.t. Alleen-schrijven, wordt gebruikt om de z-waarde (of dieptewaarde) te herschrijven die in de hoekpuntshader is geschreven.	6	N.v.t.	N.v.t.
Tokens 1024 2048		200	

De nieuwste AGAL Mini Assembler is hier beschikbaar.